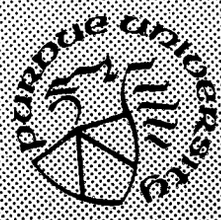
1-1-1988

Fast Algorithms for the Real Discrete Fourier Transform

O. K. Ersoy
Purdue University

N. C. Hu
Purdue University



Fast Algorithms for The Real Discrete Fourier Transform

O. K. Ersoy
N. C. Hu

TR-EE-88-5
1988

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

CONTENTS

ABSTRACT

Fast algorithms for the computation of the real discrete Fourier transform (RDFT) are discussed. Implementations based on the RDFT are always efficient whereas the implementations based on the DFT are efficient only when signals to be processed are complex. The fast real Fourier (FRFT) algorithms discussed are the radix-2 decimation-in-time (DIT), the radix-2 decimation-in-frequency (DIF), the radix-4 DIT, the split-radix DIT, the split-radix DIF, the prime-factor, the Rader prime, and the Winograd FRFT algorithms.

I. INTRODUCTION

The discrete Fourier transform (DFT) is one of the most important tools in signal processing. The DFT of a signal $x(\cdot)$ of size N is given by

$$f(n) = \sum_{k=0}^{N-1} x(k) \exp(-j2\pi nk/N) \tag{1.1}$$

The DFT can be considered to be the approximation of the complex Fourier transform (CFT) given by

$$x(f) = \int_{-\infty}^{\infty} x(t) \exp(-j2\pi tf)dt \tag{1.2}$$

The DFT and the CFT are well-suited when the signals are complex. When the signals are real, it may be more advantageous to use real transforms. The real discrete Fourier transform (RDFT) is given by [1]

$$y(n) = \sum_{k=0}^{N-1} x(k) \cos\left(\frac{2\pi nk}{N} + \theta(n)\right) \tag{1.3}$$

where

$$\theta(n) = \begin{cases} 0 & 0 \leqq n \leqq N_2 \\ \dfrac{\pi}{2} & N_2 < n < N\,; \end{cases} \tag{1.4}$$

and N_2 equals N/2 when N is even, and (N−1)/2 when N is odd.

The inverse RDFT is given by

$$x(n) = \frac{1}{N_2} \sum_{k=0}^{N-1} y(k)v(k)\cos\left[\frac{2\pi nk}{N} + \theta(k)\right] \tag{1.5}$$

where

$$v(k) = \begin{cases} 1 & k \neq 0,\ N_2 \\ \dfrac{1}{2} & k = 0,\ N_2\ . \end{cases} \tag{1.6}$$

The RDFT can be considered to be the approximation of the real Fourier transform (RFT), given by

$$y(f) = \int_{-\infty}^{\infty} x(t) \cos\left(2\pi tf + \theta(f)\right) \tag{1.7}$$

where

$$\theta(f) = 0 \quad f \geqq 0$$
$$= \pi/2 \quad f < 0. \tag{1.8}$$

In the literature, numerous applications have been traditionally expressed in terms of the DFT. Especially when the signals are real, the complex arithmetic of the DFT is not well-suited. Two approaches to come around such disadvantages have been to develop algorithms for real-valued signals to be used together with the DFT, or the use of some real trigonometric transforms such as the discrete cosine, sine and Hartley transforms [2], [3], [4]. The very power of the DFT has caused the negligence of the RDFT.

It can be easily shown that the RDFT matrix consists of the eigenvectors of a circularly symmetric covariance matrix. Consequently, the RDFT is ideal for the implementation of zero-phase or linear-phase FIR filters. The RDFT gives better performance than the DFT in applications such as signal representation and nonlinear matched filtering [1], [5]. It is very efficient for the computation of real circular convolution [6]. When the other discrete trigonometric transforms are expressed in terms of the RDFT, the best fast algorithms for the RDFT also correspond to the best fast algorithms for the other transforms in terms of the number of operations [7]. The scrambled RDFT (SRDFT) gives basically the same performance in image compression as the discrete cosine transform (DCT) with much fewer number of operations [8]. A number of claims such as Walsh-filtering being more efficient than Fourier-filtering can be shown to be untrue once the RDFT is utilized [8].

The purpose of this article is to discuss a number of fast algorithms to compute the RDFT, by treating the RDFT on its own, rather than going through the intermediate processing with the DFT. The fast algorithms for the computation of the RDFT, to be discussed in the following sections, will be referred to as FRFT.

The RDFT actually corresponds to the first class out of four possible classes of the generalized real discrete Fourier transforms (GRDFT) [9]. For this reason, the RDFT will also be denoted by R_1. The other three classes are denoted by R_2, R_3, R_4. Each class R_i consists of a combination of a discrete cosine C_i and a sine transform S_i. R_2 will be needed in the discussion of the fast algorithms. R_2 can be expressed as

$$z(n) = \sum_{k=0}^{N-1} x(k) \cos\left[\frac{2\pi n(k+1/2)}{N} + \phi(n)\right] \tag{1.9}$$

where

$$\phi(n) = 0 \quad 0 \leqq n < N_2$$
$$= \frac{\pi}{2} \quad N_2 \leqq n < N \tag{1.10}$$

Let R_1 and R_2 also denote the matrices for the two transforms. The relationship

between R_1 and R_2 can be written as

$$R_2 = TR_1 \tag{1.11}$$

where the elements of the matrix T are given by

$$T(0,0) = 1 \tag{1.12a}$$

$$T(N_2,N_2) = -1 \tag{1.12b}$$

and for $0 < n < N_2$,

$$T(n,n) = -T(N-n,N-n) = \cos\frac{\pi n}{N} \tag{1.12c}$$

$$T(N-n,n) = T(n,N-n) = -\sin\frac{\pi n}{N} \tag{1.12d}$$

When N is odd, Eq. (1.12b) is removed and n also equals N_2 in Eqs. (1.12c) and (1.12d).

Once the RDFT is computed, other transforms can easily be obtained from it. For example, the DFT coefficients f(n) are given by

$$f(0) = y(0) \tag{1.13a}$$

$$f(N_2) = y(N_2)$$

and for $n \neq 0,N_2$,

$$\begin{bmatrix} f(n) \\ f(N-n) \end{bmatrix} = \begin{bmatrix} 1 & -j \\ 1 & j \end{bmatrix} \begin{bmatrix} y(n) \\ y(N-n) \end{bmatrix} \tag{1.13c}$$

Similarly, the discrete Hartley transform (DHT) coefficients $h(\cdot)$ can be written as

$$h(0) = y(0) \tag{1.14a}$$

$$h(N_2) = y(N_2) \tag{1.14b}$$

and for $n \neq 0,N_2$,

$$\begin{bmatrix} h(n) \\ h(N-n) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} y(n) \\ y(N-n) \end{bmatrix} \tag{1.14c}$$

In the succeeding sections, the basic building block for the fast algorithms will be observed to be the Givens' plane rotation, in the form

$$\begin{bmatrix} a' \\ b' \end{bmatrix} = \begin{bmatrix} \cos\frac{2\pi n}{M} & -\sin\frac{2\pi n}{M} \\ \sin\frac{2\pi n}{M} & \cos\frac{2\pi n}{M} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \tag{1.15}$$

This operation will be indicated by a cross with the label G_n in the signal flow-diagrams, as shown in Fig. 1. When the label is missing, it signifies an add (upper address) and subtract (lower address) operation. The number of additions and multiplications in each algorithm will be denoted by A(N) and M(N), respectively.

2. THE RADIX-2 DECIMATION-IN-TIME (DIT) FRFT ALGORITHM

For N a power of 2, Eq. (1.3) can be written as

$$y(n) = y_1(n) + y_2(n) \tag{2.1}$$

where

$$y_1(n) = \sum_{k=0}^{N_2-1} x(2k)\cos\left[\frac{2\pi nk}{N_2} + \theta(n)\right] \tag{2.2}$$

and

$$y_2(n) = \sum_{k=0}^{N_2-1} x(2k+1)\cos\left[\frac{2\pi n(k+1/2)}{N_2} + \phi(n)\right] \tag{2.3}$$

Let N_4 be N/4. Using the properties of cosines and sines, Eqs. (2.2) and (2.3) can be interpreted to be R_1 and R_2 of size N_2, respectively, with the following relations:

$$y(0) = y_1(0) + y_2(0) \tag{2.4a}$$

$$y(N_2) = y_1(0) - y_2(0) \tag{2.4b}$$

$$y(N_4) = y_1(N_4) \tag{2.4c}$$

$$y(3N_4) = -y_2(N_4) \tag{2.4d}$$

and, for $0 < n < N_4$,

$$y(n) = y_1(n) + y_2(n) \tag{2.4e}$$

$$y(N_2-n) = y_1(n) - y_2(n) \tag{2.4f}$$

$$y(N_2 + n) = -y_1(N_2 - n) - y_2(N_2 - n) \tag{2.4g}$$

$$y(N - n) = y_1(N_2 - n) - y_2(N_2 - n) \tag{2.4h}$$

The procedure described above can be continued iteratively until reaching R_1 and R_2 of size 2.

At stage k, R_1 and R_2 of size 2^k is computed from R_1 and R_2 of size 2^{k-1} of the previous stage. R_2 of size M is computed from R_1 of size M, using Eq. (1.12).

The algorithm obtained above in terms of R_1 and R_2 gives the same results as the pruning of the radix-2 DIT FFT [10]. The program provided in Ref. [10] is rewritten in Appendix B in terms of the equations above, in order to show the recursive use of R_1 and R_2 and the equivalence between the two procedures.

Fig. 2 shows the signal-flow graph for the radix-2 DIT FRFT when $N = 16$. It is observed that the number of stages required is $5 \neq \log_2 N$, due to the fact that each cross in Fig. 2 signifies real operations with the basic building block of Givens' plane rotation instead of a complex butterfly. Because the permutations at each stage, which are present in the radix-2 DIT FFT, is avoided, the output comes out in permuted order. The minus signs at some nodes indicate that the output of the node is negated. This is a consequence of Eq. (2.4g).

The number of operations in the radix-2 DIT FRFT can be easily shown to be

$$A(N) = \frac{7N}{4}(\log_2 N - 2) + 6 \tag{2.5a}$$

$$M(N) = \frac{3N}{4}(\log_2 N - 2) - N + 4 \tag{2.5b}$$

3. THE RADIX-2 DECIMATION-IN-FREQUENCY (DIF) FRFT ALGORITHM

For $0 \leqq n < N_2$, let

$$x_1(n) = x(n) + x(n + N_2) \tag{3.1a}$$

$$x_0(n) = x(n) - x(n + N_2) \tag{3.1.b}$$

Then, Eq. (1.3) can be written as

$$y(2n) = \sum_{k=0}^{N_2-1} x_1(k)\cos\left[\frac{2\pi nk}{N_2} + \theta(2n)\right] \tag{3.2}$$

$$y(2n+1) = \sum_{k=0}^{N_2-1} x_0(k)\cos\left[\frac{2\pi k(n+1/2)}{N_2} + \theta(2n+1)\right] \tag{3.3}$$

Eqs. (3.2) and (3.3) are R_1 and R_3 of size N_2, respectively [9].

R_3 can be computed fast in two ways. The first method is to compute it in terms of the discrete cosine and sine transforms C_3 and S_3[9]. C_3 and S_3 are, in turn, computed in terms of R_1.

In the second method, for $0 \leqq n < N_2$, R_3 is expressed as follows:

$$y(2n+1) = \sum_{k=0}^{N_2-1} \left[x_0(k)\cos\frac{\pi k}{N_2}\right]\cos\frac{2\pi nk}{N_2} - \sum_{k=0}^{N_2-1} \left[x_0(k)\sin\frac{\pi k}{N_2}\right]\sin\frac{2\pi nk}{N_2} \tag{3.4a}$$

$$y(N-2n-1) = \sum_{k=0}^{N_2-1} \left[x_0(k)\sin\frac{\pi k}{N_2}\right]\cos\frac{2\pi nk}{N_2} + \sum_{k=0}^{N_2-1} \left[x_0(k)\cos\frac{\pi k}{N_2}\right]\sin\frac{2\pi nk}{N_2} \tag{3.4b}$$

Eq. (3.4) involves the computation of 2 R_1's. Thus, a total of 3 R_1's instead of 2 R_1's are needed at each stage, as in the case of the first method. This makes the DIF FRFT inefficient and is not pursued further.

4. THE RADIX-4 DECIMATION-IN-TIME FRFT ALGORITHM

R_1 of size 4 can be written as follows:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & -1 & 0 \\ 1 & -1 & 1 & -1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{4.1}$$

Eq. (4.1) can be computed with only 6 additions. When N is a power of 4, a fast algorithm can be built up from size-4 R_1, with less number of operations than with a radix-2 DIT FRFT.

For $0 \leqq n < N_4$, let

$$x_a(n) = x(4n) \tag{4.2a}$$

$$x_b(n) = x(4n+1) \tag{4.2b}$$

$$x_c(n) = x(4n+2) \tag{4.2c}$$

$$x_d(n) = x(4n+3) \tag{4.2d}$$

Then, Eq. (1.3) can be written as

$$y(n) = \sum_{k=0}^{N_4-1} x_a(k)\cos\left[\frac{2\pi nk}{N_4} + \theta(n)\right] + \cos\frac{2\pi n}{N}\sum_{k=0}^{N_4-1} x_b(k)\cos\left[\frac{2\pi nk}{N_4} + \theta(n)\right]$$

$$- \sin\frac{2\pi n}{N}\sum_{k=0}^{N_4-1} x_b(k)\sin\left[\frac{2\pi nk}{N_4} + \theta(n)\right] + \cos\frac{4\pi n}{N}\sum_{k=0}^{N_4-1} x_c(k)\cos\left[\frac{2\pi nk}{N_4} + \theta(n)\right]$$

$$- \sin\frac{4\pi n}{N}\sum_{k=0}^{N_4-1} x_c(k)\sin\left[\frac{2\pi nk}{N_4} + \theta(n)\right] + \cos\frac{6\pi n}{N}\sum_{k=0}^{N_4-1} x_d(k)\cos\left[\frac{2\pi nk}{N_4} + \theta(n)\right]$$

$$- \sin\frac{6\pi n}{N}\sum_{k=0}^{N_4-1} x_d(k)\sin\left[\frac{2\pi nk}{N_4} + \theta(n)\right] \tag{4.3}$$

Let the R_1's of the data sequences $x_a(\cdot)$, $x_b(\cdot)$, $x_c(\cdot)$ and $x_d(\cdot)$ of size N_4 be denoted by $y_a(\cdot)$, $y_b(\cdot)$, $y_c(\cdot)$ and $y_d(\cdot)$, respectively. For $0 < n < N_8$, N_8 being $N/8$, the following will be defined,

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \cos\frac{2\pi n}{N} & -\sin\frac{2\pi n}{N} \\ \sin\frac{2\pi n}{N} & \cos\frac{2\pi n}{N} \end{bmatrix} \begin{bmatrix} y_b(n) \\ y_b(N_4-n) \end{bmatrix} \tag{4.4}$$

$$\begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} \cos\frac{4\pi n}{N} & -\sin\frac{4\pi n}{N} \\ \sin\frac{4\pi n}{N} & \cos\frac{4\pi n}{N} \end{bmatrix} \begin{bmatrix} y_c(n) \\ y_c(N_4-n) \end{bmatrix} \tag{4.5}$$

$$\begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} \cos\frac{6\pi n}{N} & -\sin\frac{6\pi n}{N} \\ \sin\frac{6\pi n}{N} & \cos\frac{6\pi n}{N} \end{bmatrix} \begin{bmatrix} y_d(n) \\ y_d(N_4-n) \end{bmatrix} \tag{4.6}$$

Eqs. (4.4) - (4.6) are plane rotations which can be implemented in 3 multiplications and 3 additions. However, when n=N/16, Eq. (4.5) costs 2 multiplications and 2 additions.

Eq. (4.3) can be written as

$$y(0) = \left[y_a(0) + y_c(0)\right] + \left[y_b(0) + y_d(0)\right] \tag{4.7a}$$

$$y(N_2) = \left[y_a(0) + y_c(0)\right] - \left[y_b(0) + y_d(0)\right] \tag{4.7b}$$

$$y(N_8) = y_a(N_8) + \frac{1}{\sqrt{2}}\left[y_b(N_8) - y_d(N_8)\right] \tag{4.7c}$$

$$y(3N_8) = y_a(N_8) - \frac{1}{\sqrt{2}}\left[y_b(N_8) - y_d(N_8)\right] \tag{4.7d}$$

$$y(N_4) = y_a(0) - y_c(0) \tag{4.7e}$$

$$y(3N_4) = y_b(0) - y_d(0) \tag{4.7f}$$

$$y(5N_8) = -y_c(N_8) + \frac{1}{\sqrt{2}}\left[y_b(N_8) + y_d(N_8)\right] \tag{4.7g}$$

$$y(7N_8) = y_c(N_8) + \frac{1}{\sqrt{2}}\left[y_b(N_8) + y_d(N_8)\right] \tag{4.7h}$$

and, for $0 < n < N_8$,

$$y(n) = \left[y_a(n) + c\right] + (a+e) \tag{4.7i}$$

$$y(N_2-n) = \left[y_a(n) + c\right] - (a + e) \tag{4.7j}$$

$$y(3N_4-n) = \left[y_a(N_4-n) - d\right] + (a - e) \tag{4.7k}$$

$$y(3N_4 + n) = -\left[y_a(N_4-n) - d\right] + (a - e) \tag{4.7l}$$

$$y(N_4 - n) = \left[y_a(n) - c\right] + (b - f) \tag{4.7m}$$

$$y(N_4 + n) = \left[y_a(n) - c\right] - (b - f) \tag{4.7n}$$

$$y(N_2 + n) = -\left[y_a(N_4 - n) + d\right] + (b + f) \tag{4.7o}$$

$$y(N - n) = \left[y_a(N_4 - n) + d\right] + (b + f) \tag{4.7p}$$

The complexity of Eq. (4.7) is 10 multiplications and 36 additions if $n = N/16$, 11 multiplications and 37 additions otherwise. Using these numbers, it can be easily shown that the number of operations in the radix-4 DIT FRFT algorithm are given by the following:

$$A(N) = \frac{25N}{8}\log_4 N - (67N - 112)/24 \tag{4..8a}$$

$$M(N) = \frac{9N}{8}\log_4 N - (43N - 64)/24 \tag{4.8b}$$

A Fortran program for the radix-4 DIT FRFT is provided in Appendix C.

5. THE SPLIT-RADIX DECIMATION-IN-TIME FRFT ALGORITHM

In the split-radix algorithm, the even indexed and the odd-indexed parts of the signal sequence at each stage are represented by a radix-2 and a radix-4 algorithm, respectively [11]. For this purpose, Eq. (1.3) can be written as

$$y(n) = \sum_{k=0}^{N_2-1} x(2k)\cos\left(\frac{2\pi nk}{N_2} + \theta(n)\right) + \cos\frac{2\pi n}{N}\sum_{k=0}^{N_4-1} x_b(k)\cos\left(\frac{2\pi nk}{N_4} + \theta(n)\right)$$

$$- \sin\frac{2\pi n}{N}\sum_{k=0}^{N_4-1} x_b(k)\sin\left(\frac{2\pi nk}{N_4} + \theta(n)\right) + \cos\frac{6\pi n}{N}\sum_{k=0}^{N_4-1} x_d(k)\cos\left(\frac{2\pi nk}{N_4} + \theta(n)\right) \qquad (5.1)$$

$$- \sin\frac{6\pi n}{N}\sum_{k=0}^{N_4-1} x_d(k)\sin\left(\frac{2\pi nk}{N_4} + \theta(n)\right)$$

where $x_b(\cdot)$ and $x_d(\cdot)$ are defined by Eqs. (4.2b) and (4.2d), respectively.

Similar to the radix-4 algorithm, Eq. (5.1) can be written in 8 parts. Using $y_1(\cdot)$ defined by Eq. (2.2), and $y_b(\cdot)$, and $y_d(\cdot)$ defined in Sec. 4, Eq. (5.1) becomes:

$$y(0) = y_1(0) + \left[y_b(0) + y_d(0)\right] \qquad (5.2a)$$

$$y(N_2) = y_1(0) - \left[y_b(0) + y_d(0)\right] \qquad (5.2b)$$

$$y(N_8) = y_1(N_8) + \frac{1}{\sqrt{2}}\left[y_b(N_8) - y_d(N_8)\right] \qquad (5.2c)$$

$$y(3N_8) = y_1(N_8) - \frac{1}{\sqrt{2}}\left[y_b(N_8) - y_d(N_8)\right] \qquad (5.2d)$$

$$y(N_4) = y_1(N_4) \qquad (5.2e)$$

$$y(3N_4) = -y_b(0) + y_d(0) \qquad (5.2f)$$

$$y(5N_8) = -y_1(3N_8) + \frac{1}{\sqrt{2}}\left[y_b(N_8) + y_d(N_8)\right] \qquad (5.2g)$$

$$y(7N_8) = y_1(3N_8) + \frac{1}{\sqrt{2}}\left[y_b(N_8) + y_d(N_8)\right] \qquad (5.2h)$$

and, for $0 < n < N_8$, in terms of a, b, e, f defined by Eqs. (4.4), (4.6),

$$y(n) = y_1(n) + (a + e) \qquad (5.2i)$$

$$y(N_2 - n) = y_1(n) - (a + e) \qquad (5.2j)$$

$$y(3N_4 - n) = y_1(n) + (a - e) \tag{5.2k}$$

$$y(3N_4 + n) = -y_1(n) + (a - e) \tag{5.2l}$$

$$y(N_4 - N) = y_1(N_4 - n) + (b - f) \tag{5.2m}$$

$$y(N_4 + n) = y_1(N_4 - n) - (b - f) \tag{5.2n}$$

$$y(N_2 + n) = -y_1(N_2 - n) + (b + f) \tag{5.2o}$$

$$y(N - n) = y_1(N_2 - n) + (b + f) \tag{5.2p}$$

The algorithm obtained above in terms of R_1 gives the same results as the pruning of the split-radix DIT FFT algorithm. The program provided in Ref. [10] is rewritten in Appendix D in terms of the equations above, in order to show the recursive use of R_1 and the equivalence between the two procedures.

Fig. 3 shows the signal-flow diagram for the split-radix DIT FRFT when N=16. The minus signs at some nodes indicate that the output of the node is negated. This is a consequence of Eqs. (5.2g), (5.2l) and (5.2o).

The number of operations in the split-radix DIT FRFT algorithm can be shown to be [12]

$$A(N) = \frac{N}{2}\left[3\log_2 N - 5\right] + 4 \tag{5.3a}$$

$$M(N) = \frac{N}{2}\left[\log_2 N - 3\right] + 2 \tag{5.3b}$$

6. THE SPLIT-RADIX DECIMATION-IN-FREQUENCY FRFT ALGORITHM

Eq. (1.3) can also be written as as

$$\left.\begin{aligned} X_1(n) &= \sum_{k=0}^{N-1} x(k)\cos\frac{2\pi nk}{N} \\ X_0(n) &= \sum_{k=0}^{N-1} x(k)\sin\frac{2\pi nk}{N} \end{aligned}\right\} \tag{6.1}$$

$X_1(\cdot)$ and $X_0(\cdot)$ are the real and the imaginary frequency components, respectively. Consider $X_1(2n)$ and $X_0(2n)$, given by

$$\left.\begin{aligned} X_1(2n) &= \sum_{k=0}^{N_2-1} x_1(k)\cos\frac{2\pi nk}{N_2} \\ X_0(2n) &= \sum_{k=0}^{N_2-1} x_1(k)\sin\frac{2\pi nk}{N_2} \end{aligned}\right\} \tag{6.2}$$

where $x_1(k)$ is defined by Eq. (3.1a). Eq. (6.2) is R_1 of size N_2.

Next the following number groups will be defined:

$$\left.\begin{aligned} I_1(N) &= (1+4n)\text{mod}N \\ I_2(N) &= (3+4n)\text{mod}N \end{aligned}\right\} \; n = \text{integer} \tag{6.3}$$

Consider odd indices m in the left-hand side of Eq. (1.3). If m belongs to $I_1(N)$, it can be written as $1+4l$, and if it belongs to $I_2(N)$, it can be changed to N−m since $\cos 2\pi m/N = \cos 2\pi(N-m)N$ and $\sin 2\pi m/N = -\sin 2\pi(N-m)/N$. Thus, $X_1(m)$ and $X_0(m)$ with m odd can always be written as $X_1(4l+1)$ and $X_0(4l+1)$. These are related to $y(\cdot)$ by

$$\begin{aligned} y(n) &= X_1(n) && n\in I_1(N) \\ &= X_1(N-n) && n\in I_2(N) \qquad 0 < n < N_2 \\ y(N-n) &= X_0(n) && n\in I_1(N) \\ &= -X_0(N-n) && n\in I_2(N) \end{aligned} \tag{6.4}$$

Let

$$\begin{bmatrix} x'(k) \\ x''(k) \end{bmatrix} = \begin{bmatrix} \cos\frac{2\pi k}{N} & -\sin\frac{2\pi k}{N} \\ \sin\frac{2\pi k}{N} & \cos\frac{2\pi k}{N} \end{bmatrix} \begin{bmatrix} u'(k) \\ u''(k) \end{bmatrix} \tag{6.5}$$

where

$$\left.\begin{aligned} u'(k) &= x(k) - x(k+N_2) \\ u''(k) &= x(k+N_4) - x(k+3N_4) \end{aligned}\right\} \tag{6.6}$$

Also let

$$\left.\begin{aligned} X_1'(n) &= \sum_{k=0}^{N_4-1} x'(k)\cos\frac{2\pi nk}{N_4} \\ X_0'(n) &= \sum_{k=0}^{N_4-1} x'(k)\sin\frac{2\pi nk}{N_4} \end{aligned}\right\} \tag{6.7}$$

$$\left.\begin{aligned} X_1''(n) &= \sum_{k=0}^{N_4-1} x''(k)\cos\frac{2\pi nk}{N_4} \\ X_0''(n) &= \sum_{k=0}^{N_4-1} x''(k)\sin\frac{2\pi nk}{N_4} \end{aligned}\right\} \tag{6.8}$$

Eqs. (6.7) and (6.8) are R_1 of size N_4.

Now it can be easily shown that $X_1(\cdot)$ and $X_0(\cdot)$ for odd index are given by

$$X_1(1) = X_1'(0) \tag{6.9a}$$

$$X_1(1+N_2) = X_1'(N_8) \tag{6.9b}$$

$$\left.\begin{aligned} X_1(4n+1) &= X_1'(n) - X_0''(n) \\ X_1(N-4n+1) &= X_1'(n) + X_0''(n) \end{aligned}\right\} \quad 0 < n < N_8 \tag{6.9c}$$

$$X_0(1) = X_1''(0) \tag{6.10a}$$

$$X_0(1+N_1) = X_1''(N_8) \tag{6.10b}$$

$$\left.\begin{aligned} X_0(4n+1) &= X_1''(n) + X_0'(n) \\ X_0(N-4n+1) &= X_1''(n) - X_0'(n) \end{aligned}\right\} \quad 0 < n < N_8 \tag{6.10c}$$

Eqs. (6.2), (6.9) and (6.10) show that R_1 of size N is to be computed via R_1 of sizes N_2 and N_4. The same algorithm is applied recursively to successive smaller R_1's such that R_1 of size 2 is obtained as an add/subtract operation in the end.

The number of operations in the split-radix DIF FRFT algorithm can be easily shown to be equal to the number of operations in the split-radix DIT FRFT algorithm.

The Fortran program for the split-radix DIF FRFT algorithm is provided in Appendix E. The pertinent data structures are discussed in the next section.

7. DATA STRUCTURES FOR THE SPLIT-RADIX DECIMATION-IN-FREQUENCY FRFT ALGORITHM

If one neglects, for the moment, the additions defined by Eqs. (6.9) and (6.10), a regular signal-flow diagram is obtained for the implementation of the split-radix DIF FRFT algorithm. The signal-flow diagram for $N = 16$ is shown in Fig. 4.

The add/subtract operations defined by Eqs. (6.9) and (6.10) are shown as dotted lines in the last stage of Fig. 4. When these operations are neglected, the resulting signal-flow diagram will be referred to as the regular structure.

The Fortran program for the split-radix DIF FRFT algorithm is given in Appendix E. This program was developed by constructing a few simple rules with regard to the regular structure, and the additions of Eqs. (6.9), (6.10). The rules for the regular structure will be discussed first.

The even-indexed terms are computed according to Eq. (6.2), requiring the additions of Eq. (3.1a), and the odd-indexed terms are computed according to Eqs. (6.7) - (6.10), requiring the subtractions of Eq. (6.6). The additions and subtractions dictated by Eqs. (3.1a) and (6.6) will be referred to as A. For example, the first stage of Fig. 4 is A.

The addresses involved in A in a particular stage are divided into upper and lower halves in the succeeding stage; the upper half is still A. The lower half requires the computation of Givens' plane rotations, given by Eq. (6.5). These operations will be referred to as P. For example, the lower half of the second stage of Fig. 4 is P whereas the upper half is A. Thus, we can construct the following rule:

Rule 1. A is always followed by A and P. This structure belonging to two successive stages will be referred to as AAP.

The addresses involved in P in a particular stage are divided into upper and lower halves in the succeeding stage in order to start the computation of 2 R_1's according to Eqs. (6.7) and (6.8). This means the two halves succeeding P should be A, leading to the second rule:

Rule 2. P is always followed by A and A. This structure belonging to two successive stages will be referred to as PAA.

Combining rules 1 and 2, one also reaches the following conclusions:

Rule 3. AAP is always followed by AAP, PAA, AAP, AAP.

Rule 4. PAA is always followed by AAP, PAA, AAP, PAA.

The block size of AAP and PAA (the number of addresses involved) will be called S(AAP) and S(PAA), respectively. Both S(AAP) and S(PAA) are powers of 4.

Rules 3 and 4 allow programming in terms of 2 stages at a time, as done in the appendix. These rules also lead to the conclusion that it is necessary to differentiate between the cases of $N = 2^L$, L odd and L even, as follows:

The Case of L Odd:

The first stage is implemented as A. The succeeding stages are implemented two at a time, as AAP and PAA, according to Rules 3 and 4. This process is continued until and including the block sizes S(AAP) and S(PAA) equal to 4.

For example, Fig. 5 shows the block diagram in terms of A, AAP and PAA operations when $N = 32$. The numbers in parenthesis are the block sizes.

The Case of L Even:

The stages are implemented two at a time, as AAP and PAA according to Rules 3 and 4. This process is continued until and including the block sizes S(AAP) and S(PAA) equal to 4.

For example, Fig. 6 shows the block diagram in terms of AAP and PAA operations when $N = 64$.

Next we will discuss the add/subtract operations due to Eqs. (6.9) and (6.10), which exist for $N \geqq 16$. These operations, which will be referred to as IAS of size 2M, combine the results of 2 R_1's of size M, requiring (M-2) pairs of add/subtract operations. In order to determine their locations, a sequence of control numbers (SCN) is devised as follows:

The first IAS of size 8 occurs at address $8 + 2$ when $N = 16$, as seen in Fig. 4. Reasoning through the successive smaller R_1's imbedded in a single large R_1, it is easy to find out that the starting addresses of IAS of size 8 are 2 plus the following numbers: 8, 40, 56, 72, 104, 136, 168, 184, 200.

SCN will be defined as the above sequence divided by 8, as shown below for $N \leqq 512$: SCN = [1, 5, 7, 9, 13, 17, 21, 23, 25, 29, 31, 33, 37, 39, 41, 49, 53, 55, 57, 61].

A careful study of signal flow in the split-radix DIF FRFT algorithm also leads to the following rules:

Rule 5. IAS of size $M' = 2M$ always starts at addresses given by M·SCN. For example, when $N = 512$, IAS of size 16 starts at addresses [16, 80, 112, 144, 208, 272, 336, 368, 400, 464, 496), which is $16 \cdot [1, 5, 7, 9, 13, 17, 21, 21, 23, 25, 29]$.

Rule 6. The starting addresses of PAA's of size S(PAA) are at SCN · S(PAA). For example, PAA of size 16 in Fig. 3 starts at address 16, which is 16·1. The starting addresses of PAA's of size 4 are [4, 20, 28, 36, 52], which is $4 \cdot [1, 5, 7, 9, 13]$.

The output results are in permuted order, as seen in Fig. 4. Unscrambling of the permuted sequence can best be done with a look-up table, which is provided in Table 2 for $N \leqq 256$. When $N < 256$, $M = 256/N$ is used to divide the first N numbers in this table in order to find the permutation sequence corresponding to N.

Some of the output results also have a negative sign, due to Eq. (6.4). A careful study of the signal-flow diagram indicates that these occur at the last K outputs where

$$K = \sum_{i=1}^{L} \frac{N}{4^i}, \tag{7.1}$$

L being $\log_4 N$ when N is a power of 4, and $\log_4(N/2)$ otherwise.

8. THE PRIME-FACTOR FRFT ALGORITHM

In the Good-Thomas prime-factor algorithm [12], [13], the transform length N equals $N_x N_y$, and N_x, N_y are prime to each other. Consequently, the input index n, $0 \leqq n < N-1$ can be decomposed into two indices n_x and n_y, $0 \leqq n_x < N_x$, $0 \leqq n_y < N_y$, using the Chinese remainder theorem, in the form [14]

$$n = n_x M_y N_y + n_y M_x N_x \tag{8.1}$$

where

$$n_x = n \bmod N_x \tag{8.2}$$

$$n_y = n \bmod N_y \tag{8.3}$$

M_x, M_y are integers satisfying

$$M_x N_x + M_y N_y = 1 \bmod N \tag{8.4}$$

The input index k is written as

$$k = N_y k_x + N_x k_y \tag{8.5}$$

where

$$k_x = (M_y \bmod n_x) k \bmod n_x \tag{8.6}$$

$$k_y = (M_x \bmod n_y) k \bmod n_y \tag{8.7}$$

Using the equations above, Eq. (6.1) can be written as

$$X_1(n_x,n_y) = \sum_{k_x=0}^{N_x-1}\sum_{k_y=0}^{N_y-1} x(k_x,k_y)\cos\left[2\pi\left(\frac{n_xk_x}{N_x}+\frac{n_yk_y}{N_y}\right)\right] \tag{8.8a}$$

$$X_0(n_x,n_y) = \sum_{k_x=0}^{N_x-1}\sum_{k_y=0}^{N_y-1} x(k_x,k_y)\sin\left[2\pi\left(\frac{n_xk_x}{N_x}+\frac{n_yk_y}{N_y}\right)\right] \tag{8.8b}$$

where $X_1(n_x,n_y)$ and $X_0(n_x,n_y)$ equal $X_1(n)$ and $X_0(n)$, respectively, n given by Eq. (8.1); similarly, $x(k_x,k_y)$ equals x(k), k given by Eq. (8.5).

The 2-D RDFT of size N_xxN_y is given by [15]

$$y(n_x,n_y) = \sum_{k_x=0}^{N_x-1}\sum_{k_y=0}^{N_y-1} x(k_x,k_y)\cos\left[\frac{2\pi k_x n_x}{N_x}+\theta(n_x)\right]\cos\left[\frac{2\pi n_y k_y}{N_y}+\theta(n_y)\right] \tag{8.9}$$

The relationship between $y(n_x,n_y)$ and $X_1(n_x,n_y)$, $X_0(n_x,n_y)$ is shown in Table 2. When N_x and/or N_y is odd, the rows of Table 2 containing $N_x/2$ and/or $N_y/2$ are to be removed. $X_1(n_x,n_y)$ and $X_0(n_x,n_y)$ are also the real and the imaginary parts of the 2-D DFT. Thus, the relationship between the 2-D DFT and the 2-D RDFT is also given by Table 2.

The fast computation of the 2-D RDFT can be achieved in a number of ways, such as by using the 1-D FRFT, first along the rows, and then along the columns of x (•,•), or vice versa.

The procedure for the prime-factor FRFT is as follows:

A. The 1-D signal x(•) is converted into the 2-D signal x(•,•), using Eqs. (8.4) - (8.7).

B. The 2-D RDFT y(•,•) of x(•,•) is computed.

C. $X_1(•,•)$ and $X_0(•,•)$ are computed from y(•,•), using Table 2.

D. $X_1(•,•)$ and $X_0(•,•)$ are converted to 1-D y (•), using Eqs. (8.1) - (8.4).

When the number of relatively prime factors of N are more than 2, the procedure above can be easily extended to convert the 1-D RDFT to a multi-dimensional (M-D) RDFT and to recombine the results back to 1-D in the end.

The number of operations in the prime-factor FRFT algorithm can be shown to be the following:

$$M(N) = N_xM(N_y) + N_yM(N_x) \tag{8.10a}$$

N_x even, N_y odd:

$$A(N) = N_xA(N_y) + N_yA(N_x) + 4(N_x/2-1)((N_y+1)/2-1) \tag{8.10b}$$

N_x and N_y odd:

$$A(N) = N_x A(N_y) + N_y A(N_x) + 4((N_x+1)/2-1)((N_y+1)/2-1) \tag{8.10c}$$

$A(N_x), A(N_y), M(N_x), M(N_y)$ depend on the particular FRFT algorithm.

9. THE RADER PRIME FRFT ALGORITHM

In the Rader prime algorithm [16], the number of data points N is a prime number. Then, there exists a primitive root p of N such that each positive integer less than N can be expressed as a unique power of p mod N.

Eq. (6.1) can be written, for $0 \leqq n < N_2$, as

$$X_1(0) = \sum_{k=0}^{N-1} x(k) \tag{9.1a}$$

$$X_1(p^n) - x(0) = \sum_{k=0}^{N-2} x(p^k)\cos\left(\frac{2\pi}{N}p^{k+n}\right) \tag{9.1b}$$

$$X_0(p^n) = \sum_{k=0}^{N-2} x(p^k)\sin\left(\frac{2\pi}{N}p^{k+n}\right) \tag{9.1c}$$

Let N_0 be $(N-1)/2$. Since $\cos 2\pi n/N$ equals $\cos 2\pi(n-n)/N$, and $\sin 2\pi n/N$ equals $-\sin 2\pi(N-n)/N$, Eqs. (9.1b) and (9.1c) can also be written as

$$X_1(p^n) - x(0) = \sum_{k=0}^{N_0-1} x_1(p^k)\cos\left(\frac{2\pi}{N}p^{k+n}\right) \tag{9.2b}$$

$$X_0(p^n) = \sum_{k=0}^{N_0-1} x_0(p^k)\,\mathrm{sgn}(N_0-p^k)\sin\left(\frac{2\pi}{N}p^{k+n}\right) \tag{9.2c}$$

where $\mathrm{sgn}(\cdot)$ is the sign function, and $x_1(\cdot)$, $x_0(\cdot)$ are given by Eq. (3.1). It is assumed that $x_1(l)$ and $x_0(l)$ equal $x_1(N-l)$, and $x_0(N-l)$, respectively, when l is greater than N_0.

Eqs. (9.2a) and (9.2b) are circular and skew-circular correlations of size N_0, respectively.

For example, let us consider N=5. Let c(n,N) and s(n,N) represent $\cos 2\pi n/N$ and $\sin 2\pi n/N$, respectively. Then Eq. (9.2) can be written as

$$\begin{bmatrix} X_1(1) - x(0) \\ X_1(2) - x(0) \end{bmatrix} = \begin{bmatrix} c(1,5) & c(2,5) \\ c(2,5) & c(1,5) \end{bmatrix} \begin{bmatrix} x_1(1) \\ x_1(2) \end{bmatrix} \tag{9.3a}$$

$$\begin{bmatrix} X_0(1) \\ X_0(2) \end{bmatrix} = \begin{bmatrix} s(1,5) & s(2,5) \\ s(2,5) & -s(1,5) \end{bmatrix} \begin{bmatrix} x_0(1) \\ x_0(2) \end{bmatrix} \tag{9.3b}$$

The Rader algorithm can still be used when N equals N_p^n, N_p being an odd prime [14]. In this case, there is no primitive root of order N−1, but there is an element a of order N_1 equal to $N_p^{n-1}(N_p-1)$. In order to make use of a, all the rows and columns of the transformation matrix with index i equal to 0 or having a factor containing N_p are deleted. For the remaining indices, Eq. (6.1) can be written, for $0 \leqq n < N_1$, as

$$X_1(a^n) - x(0) = \sum_{k=0}^{N_1/2-1} x_1(a^k)\cos(\frac{2\pi}{N}a^{k+n}) \tag{9.4a}$$

$$X_0(a^n) = \sum_{k=0}^{N_1/2-1} x_0(a^k)\mathrm{sgn}(N_0 - a^k)\sin(\frac{2\pi}{N}a^{k+n}) \tag{9.4b}$$

Eqs. (9.4a) and (9.4ba) are circular and skew-circular correlations of sizes $N_1/2$, respectively.

The other transform components with index containing $N_p^{m'}$, $m' < m$, as a factor can be handled by the same algorithm after reducing N by $N_p^{m'}$.

For example, let us consider N=9. Then a equals 2. Eq. (9.4) can be written as

$$\begin{bmatrix} X_1(1) - x(0) \\ X_1(2) - x(0) \\ X_1(4) - x(0) \end{bmatrix} = \begin{bmatrix} c(1,9) & c(2,9) & c(4,9) \\ c(2,9) & c(4,9) & c(1,9) \\ c(4,9) & c(1,9) & c(2,9) \end{bmatrix} \begin{bmatrix} x_1(1) \\ x_1(2) \\ x_1(4) \end{bmatrix} \tag{9.5a}$$

$$\begin{bmatrix} X_0(1) \\ X_0(2) \\ X_0(4) \end{bmatrix} = \begin{bmatrix} s(1,g) & s(2,9) & s(4,9) \\ s(2,9) & s(4,9) & -s(1,9) \\ s(4,9) & -s(1,9) & -s(2,9) \end{bmatrix} \begin{bmatrix} x_0(1) \\ x_0(2) \\ x_0(4) \end{bmatrix} \tag{9.5b}$$

The other components with index 3 can be treated as in N=3 algorithm. Thus,

$$X_1(3) - x(0) = c(1,3)x_1(3) \tag{9.5c}$$

$$X_0(3) = s(1,3)x_0(3) \tag{9.5d}$$

When N is a power of 2, the circular and skew-circular correlations are of size N/8, N/8...2. The details of this algorithm can be found in Ref. [17].

10. THE WINOGRAD FRFT ALGORITHM

In the Winograd algorithm, the circular convolutions are computed by small convolution algorithms [18]. The Winograd FRFT algorithm is similarly obtained by computing the circular and skew-circular correlations by small convolution algorithms. For example, the N=9 algorithm presented in the last section is computed by a circular and a skew-circular convolution algorithm of size 3.

The end result of this approach, as discussed below, is to factorize the transform matrix R_1 as

$$R_1 = CDA \tag{10.1}$$

where A and C are simple rectangular matrices, and D is a diagonal matrix containing the multiplicative terms.

When N consists of factors prime to each other, the Good-Thomas prime factor algorithm can be first used to convert the 1-D RDFT to a M-D RDFT, as in Sec. 8, followed by a number of simplifications to obtain a result similar to Eq. (10.1).

For the sake of simplicity, let N be N_xN_y, N_x and N_y relatively prime to each other. The result of the application of the Good-Thomas prime factor algorithm to the 1-D problem is the conversion of the problem to 2-D in the form

$$y = \left[R_1(N_x)\right] x \left[R_1(N_y)\right]^t \tag{10.2}$$

where x and y are the 2-D input, output matrices, and $\left[R_1(N_x)\right]$, $\left[R_1(N_y)\right]$ are the RDFT matrices of size N_x and N_y, respectively.

Eq. (10.2) can also be written as [14]

$$y' = \left(\left[R_1(N_y)\right] \otimes \left[R(N_x)\right]\right) x' \tag{10.3}$$

where $\otimes$ is the Kronecker-product operation; x' and y' are 1-D vectors of size N, obtained from x and y by concatenating their columns, respectively.

In turn, $\left[R_1(N_x)\right]$ and $\left[R_1(N_y)\right]$ can be written as

$$\left[R_1(N_x)\right] = C_xD_xA_x \tag{10.4}$$

$$\left[R_1(N_y)\right] = C_yD_yA_y \tag{10.5}$$

The Kronecker-product of (10.4) and (10.5) gives

$$\left[R_1(N)\right]' = C_yD_yA_y \otimes C_xD_xA_x \tag{10.6}$$

where $\left[R_1(N)\right]'$ denote $\left[R_1(N)\right]$ with permuted rows and columns, due to the concatenation procedure described earlier.

Since [14]

$$[A \otimes B][C \otimes D] = [AC] \otimes [BD], \tag{10.7}$$

Eq. (10.6) can be written as

$$[R_1(N)]' = C_T D_T A_T \tag{10.8}$$

where

$$C_T = C_y \otimes C_x \tag{10.9}$$

$$D_T = D_y \otimes D_x \tag{10.10}$$

$$A_T = A_y \otimes A_x \tag{10.11}$$

The matrices A, D, C for small size RDFT's can be easily constructed from the corresponding small size DFT's listed in the literature [14]. This is due to the fact that the matrices A and D for the DFT are real and C for the DFT has elements which are purely real or purely imaginary. Consequently, the RDFT coefficients $X_1(\cdot)$ $(X_0(\cdot))$ are obtained by setting the imaginary (real) elements in the corresponding rows of the matrix C to zero. The resulting short Winograd FRFT algorithms are given in Appendix E.

As an example, let us consider N=12. Choosing $N_x = 3$ and $N_y = 4$, the corresponding matrices are:

$$A_x = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & -1 \end{bmatrix} \tag{10.12}$$

$$D_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\frac{2\pi}{3} - 1 & 0 \\ 0 & 0 & \sin\frac{2\pi}{3} \end{bmatrix} \tag{10.13}$$

$$C_x = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10.14}$$

and

$$A_y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{10.15}$$

$$D_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10.16}$$

$$C_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10.17}$$

Eqs. (10.9) - (10.11) gives

$$A_T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 & -1 \\ 1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 \\ 0 & 1 & 1 & 0 & -1 & -1 & 0 & 1 & 1 & 0 & -1 & -1 \\ 0 & 1 & -1 & 0 & -1 & 1 & 0 & 1 & -1 & 0 & -1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \tag{10.18}$$

$$D_T = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \cos\frac{2\pi}{3}-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \sin\frac{2\pi}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \cos\frac{2\pi}{3}-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \sin\frac{2\pi}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \cos\frac{2\pi}{3}-1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sin\frac{2\pi}{3} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cos\frac{2\pi}{3}-1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sin\frac{2\pi}{3}
\end{bmatrix} \tag{10.19}$$

$$C_T = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \tag{10.20}$$

After the computations with the matrices A_T, D_T, C_T, the final 1-D RDFT output coefficients $X_1(\cdot)$ and $X_0(\cdot)$ are computed by using Table 2. This is straightforward to do by tracing the conversions from the 1-D $x(\cdot)$ to 2-D $x(\cdot,\cdot)$ according to CRT, and finally the concatenations of the columns of $x(\cdot,\cdot)$ to get $x'(\cdot)$.

11. ALGORITHMS FOR THE INVERSE RDFT AND DFT

R_1^{-1} can be computed using the same subroutine for R_1. In order to do so, let

$$x(0) = z(0) \tag{11.1a}$$

$$x(N_2) = z(N_2) \tag{11.1b}$$

$$\left.\begin{aligned} x(n) &= z(n) + z(N-n) \\ x(N-n) &= z(n) - z(N-n) \end{aligned}\right\} \quad 0 < n < N_2 \tag{11.1c}$$

and

$$u(0) = y(0) \tag{11.2a}$$

$$u(N_2) = y(N_2) \tag{11.2b}$$

$$\left.\begin{aligned} u(n) &= y(n) + y(N-n) \\ u(N-n) &= y(n) - y(N-n) \end{aligned}\right\} \quad 0 < n < N_2 \tag{11.2c}$$

Then

$$z(n) = \frac{1}{N} \sum_{k=0}^{N-1} u(k) \cos\left[\frac{2\pi nk}{N} + \theta(n)\right] \tag{11.3}$$

which is the same as Eq. (1.3) except for the normalization factor 1/N.

Eqs. (11.1) and (11.2) are exactly the same equation implemented in a single subroutine, say, comin (x,n,j). If the routine for R_1 is frft(x,N), then the calling sequence for R_1^{-1} is

```
call comin (x,N,1)
call frft (x,N)
call comin (y,N,2)
```

The flag j in comin is used to include the normalization factor 1/N in the second call to comin (j=2).

The computation of the DFT with complex input data x(•) is equivalent to computing 2 R_1's and then combining the results in the end. If f(•) is the DFT output, $y_1(\cdot)$ and $y_2(\cdot)$ are the R_1 outputs to the real and the imaginary parts of the input data, the three are related by

$$f(0) = y_1(0) + jy_2(0) \tag{11.4a}$$

$$f(N_2) = y_1(N_2) + jy_2(N_2) \tag{11.4b}$$

$$\left.\begin{aligned} f(n) &= y_1(n) + y_2(N-n) + j(y_2(n) - y_1(N-n)) \\ f(N-n) &= y_1(n) - y_2(N-n) + j(y_2(n) + y_1(N-n)) \end{aligned}\right\} \quad 0 < n < N_2 \qquad (11.4c)$$

The resulting algorithm will be referred to as R_1-FFT.

If the subroutine to implement Eq. (11.4) is called rldft(x_1,x_2,N,flag), where x_1 and x_2 are the real and imaginary parts of f at the output, and of data at the input, the calling sequence for R_1-FFT is given by

```
call frft(x1,N)
call frft(x2,N,)
call rldft(x1,x2,N,)
```

With this approach, there is no need to have a separate program for the inverse DFT (IDFT) either. In the case of the IDFT, Eq. (11.4) is modified to

$$f(0) = (y_1(0) + jy_2(0))/N$$
$$f(N_2) = (y_2(N_2) + jy_2(N_2)/N$$

$$\left.\begin{aligned} f(n) &= [y_1(n) - y_2(N-n) + j(y_2(N) + y_1(N-n))]/N \\ f(N-n) &= [y_1(n) + y_2(N-n) + j(y_2(N) - y_1(N-n))]/N \end{aligned}\right\} \quad 0 < n < N_2 \qquad (11.5)$$

In other words, in addition to scaling with 1/N, f(n) and f(N-n) are interchanged. These changes are included in rldft corresponding to flag j equal to 2. Thus, the calling sequence for the IDFT becomes

```
call frft(x1,N)
call frft(x2,N)
call rldft(x1,x2,N,2)
```

CONCLUSIONS

The fast algorithms discussed above for the computation of the real discrete Fourier transform are expected to be useful in applications, especially in order to process real signals without intermediate processing with complex signals. Many applications consist of real signals only. When complex signals are needed, the DFT computations can be achieved as in Sec. 11 without any loss of efficiency. As a matter of fact, this approach can be preferable since the real and the imaginary parts of the signal can be processed in parallel. On the other hand, when the signals are real, the computation of the RDFT by a fast implementation of the DFT is necessarily inefficient. Consequently, hardware, say, VLSI, and software implementations of Fourier processing of signals can be preferably based on the RDFT rather than the DFT. This rationale is further strengthened by the advantages of the RDFT discussed in Sec. 1.

However, there are some disadvantages of the FRFT algorithms. The radix-2 DIF FRFT algorithm is inefficient, as discussed in Sec. 3. The signal-flow diagrams for the FRFT are more difficult to understand than the FFT. If permutations are avoided in the intermediate stages, both the input and the output data are in permuted order. These issues should be further studied for possible improvements.

APPENDIX A

The Winograd small RDFT algorithms are given for n=2,3,4,5 and 7. The algorithms are in the form

$$X = CDAx$$

The matrix D is a diagonal matrix, and only the diagonal elements are given. The matrices A and C are given in full.

2-point RDFT: 0 multiplications, 2 additions

$$A = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \begin{matrix} D_0 = 1 \\ D_1 = 1 \end{matrix} \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{matrix} a_0 = x_0 + x_1 & X_0 = a_0 \\ a_1 = x_0 - x_1 & X_1 = a_1 \end{matrix}$$

3-point RDFT: $\theta = \frac{2\pi}{3}$. 1 multiplication (1 shift), 4 additions

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & -1 \end{bmatrix} \quad \begin{matrix} D_0 = 1 \\ D_1 = \cos\theta - 1 \\ D_2 = \sin\theta \end{matrix} \quad C = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{matrix} a_2 = x_1 + x_2 & X_0 = a_0 \\ a_1 = x_1 - x_2 & X_1 = -a_1/2 + a_0 \\ a_0 = x_0 + a_2 & X_2 = a_2 \cdot D_2 \end{matrix}$$

4-point RDFT: 0 multiplications, 6 additions

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \qquad \begin{aligned} D_0 &= 1 \\ D_1 &= 1 \\ D_2 &= 1 \\ D_3 &= 1 \end{aligned} \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} a_2 &= x_0 - x_2 \\ a_3 &= x_1 - x_3 \\ t_0 &= x_0 + x_2 \\ t_1 &= x_1 + x_3 \\ a_0 &= t_0 + t_1 \\ a_1 &= t_0 - t_1 \end{aligned} \qquad \begin{aligned} X_0 &= a_0 \\ X_1 &= a_2 \\ X_2 &= a_1 \\ X_3 &= a_3 \end{aligned}$$

5-point RDFT: $\theta = \dfrac{2\pi}{5}$, 5 multiplications, 13 additions

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 \end{bmatrix} \qquad \begin{aligned} D_0 &= 1 \\ D_1 &= \frac{1}{2}(\cos\theta + \cos 2\theta) - 1 \\ D_2 &= \frac{1}{2}(\cos\theta - \cos 2\theta) \\ D_3 &= \sin\theta \\ D_4 &= \sin\theta + \sin 2\theta \\ D_5 &= \sin 2\theta - \sin\theta \end{aligned} \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 & 0 \end{bmatrix}$$

$$\begin{aligned} t_0 &= x_1 + x_4 \\ t_1 &= x_2 + x_3 \\ a_4 &= x_3 - x_2 \\ a_5 &= x_1 - x_4 \\ a_1 &= t_0 + t_1 \\ a_2 &= t_0 - t_1 \\ a_3 &= a_4 + a_5 \\ a_0 &= x_0 + a_1 \end{aligned} \qquad \begin{aligned} X_0 &= a_0 \\ X_1 &= (a_0 + D_1 a_1) + D_2 a_2 = t_0 + D_2 a_2 \\ X_2 &= t_0 - D_2 a_2 \\ X_3 &= a_3 D_3 + a_5 D_5 \\ X_5 &= a_3 D_3 - a_4 D_4 \end{aligned}$$

7-point RDFT: $\theta = \dfrac{2\pi}{7}$, 8 multiplications, 30 additions

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 1 & -1 \\ 0 & 1 & 1 & -1 & 1 & -1 & -1 \\ 0 & 1 & 0 & 1 & -1 & 0 & -1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$D_0 = 1$$

$$D_1 = \frac{1}{3}(\cos\theta + \cos 2\theta + \cos 3\theta) - 1$$

$$D_2 = \frac{1}{3}(2\cos\theta - \cos 2\theta - \cos 3\theta)$$

$$D_3 = \frac{1}{3}(\cos\theta - 2\cos 2\theta - \cos\theta)$$

$$D_4 = \frac{1}{3}(\cos\theta + \cos 2\theta - 2\cos 3\theta)$$

$$D_5 = \frac{1}{3}(\sin\theta + \sin 2\theta - \sin 3\theta)$$

$$D_6 = \frac{1}{3}(2\sin\theta - \sin 3\theta + \sin 3\theta)$$

$$D_7 = \frac{1}{3}(\sin\theta - 2\sin 2\theta - \sin 3\theta)$$

$$D_8 = \frac{1}{3}(\sin\theta + \sin 2\theta + 2\sin 3\theta)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$t_0 = x_1 + x_6$$
$$t_1 = x_1 - x_6$$
$$t_2 = x_2 + x_5$$
$$t_3 = x_2 - x_5$$
$$t_4 = x_4 + x_3$$
$$t_5 = x_4 - x_3$$
$$t_6 = t_2 + t_0$$
$$a_4 = t_2 - t_0$$
$$a_2 = t_0 - t_4$$
$$a_3 = t_4 - t_2$$
$$t_7 = t_5 + t_3$$
$$a_7 = 5_5 - t_3$$
$$a_6 = t_1 - t_5$$
$$a_8 = t_3 - t_1$$
$$a_1 = t_6 + t_4$$
$$a_5 = t_7 + t_1$$
$$a_0 = x_0 + a_1$$

$$X_0 = a_0$$
$$X_1 = a_0 + a_1D_1 - a_2D_2 - a_3D_3$$
$$X_2 = a_0 + a_1D_1 - a_2D_2 - a_4D_4$$
$$X_3 = a_0 + a_1D_1 - a_3D_3 + a_4D_4$$
$$X_4 = -a_5D_5 + a_7D_7 - a_8D_8$$
$$X_5 = a_5D_5 - a_6D_6 - a_8D_8$$
$$X_6 = a_5D_5 + a_6D_6 + a_7D_7$$

APPENDIX B

```
C
C       The radix-2 decimation-in-time FRFT program
C       The output is in order:
C       [re(0),re(1),...,re(N/2),im(N/2-1),im(N/2-2),...,im(1)]
C
        SUBROUTINE FRFT(X,N)
        REAL X(1)
        M=INT(ALOG(FLOAT(N))/ALOG(2.0)+0.5)
C       Bit-reverse permutation of input data :
        J=1
        N1=N-1
        DO 104 I=1,N1
          IF(I.GE.J)GOTO 101
          XT=X(J)
          X(J)=X(I)
          X(I)=XT
101       K=N/2
102       IF(K.GE.J)GOTO 103
          J=J-K
          K=K/2
          GOTO 102
103       J=J+K
104       CONTINUE
C-------------------------------------------------------------------
C
C       DO 60 loop below is size 2 R1:
C
        DO 60 I=1,N,2
          XT=X(I)
          X(I)=XT+X(I+1)
          X(I+1)=XT-X(I+1)
60        CONTINUE
C-------------------------------------------------------------------
C
C       Computation of larger R1's:
C
          N2=1
          DO 10 K=2,M
C
```

```
C     Computation of size 2**K R1:
C
          N4=N2
          N2=2*N4
          N1=2*N2
          E=6.283185307179586/N1
          DO 20 I=1,N,N1
            XT=X(I)
C
C     X(I) is Y(0) of eqn. (2.4a):
C
            X(I)=XT+X(I+N2)
C
C     X(I+N2) is Y(N2) of eqn. (2.4b):
C
            X(I+N2)=XT-X(I+N2)
           A=E
          DO 30 J=1,N4-1
           I1=I+J
           I2=I-J+N2
           I3=I+J+N2
           I4=I-J+N1
           CC=COS(A)
           SS=SIN(A)
           A=A+E
C
C     The 7 lines below compute  T1=X(i3)*CC-X(i4)*SS,
C     T2=X(i3)*SS+X(i4)*CC ; T1 and -T2 are R2 of size 2**k ;
C     T1 is the cosine term of R2, -T2 is the sine term:
C
           P1=CC+SS
           P2=SS-CC
           P3=CC*(X(I3)+X(I4))
           P4=P1*X(I4)
           P5=P2*X(I3)
           T1=P3-P4
           T2=P3+P5
C
C     X(I4) is Y(N-n) of eqn. (2.4h):
C
```

```
            X(I4)=X(I2)+T2
C
C     X(I3) is Y(N2+n) of eqn. (2.4g):
C
            X(I3)=-X(I2)+T2
C
C     X(I2) is Y(N2-n) of eqn. (2.4f):
C
            X(I2)=X(I1)-T1
C
C     X(I1) is Y(n) of eqn. (2.4e):
C
            X(I1)=X(I1)+T1
30          CONTINUE
20        CONTINUE
10      CONTINUE
      RETURN
      END
```

APPENDIX C

```
C
C      The radix-4 decimation-in-time FRFT program
C      The output is  in order:
C      [re(0),re(1),...,re(N/2),im(N/2-1),im(N/2-2),...,im(1)]
C
       SUBROUTINE FRFT(X,N)
       REAL X(1)
       M=INT(ALOG(FLOAT(N))/ALOG(4.0)+0.5)
C
C      Bit-reverse permutation of input data:
C
       J=1
       N1=N-1
       DO 104 I=1,N1
         IF(I.GE.J)GOTO 101
         XT=X(J)
         X(J)=X(I)
         X(I)=XT
101      K=N/2
102        IF(K.GE.J)GOTO 103
         J=J-K
         K=K/2
         GOTO 102
103      J=J+K
104      CONTINUE
C ---------------------------------------------------------------
C
C      Do 60 and 61 loops below are size 4 R1:
C
70       DO 60 I=1,N,2
           XT=X(I)
           X(I)=XT+X(I+1)
           X(I+1)=XT-X(I+1)
60       CONTINUE
         DO 61 I=1,N,4
             XT=X(I)
             X(I)=XT+X(I+2)
             X(I+2)=XT-X(I+2)
61       CONTINUE
```

```
C-------------------------------------------------------------
      N2=4
      DO 10 K=2,M
         N2=N2*4
         N4=N2/4
         N8=N2/8
         E=6.283185307179586/N2
      DO 38 I=0,N-1,N2
         I1=I+1
         I2=I1+N4
         I3=I2+N4
         I4=I3+N4
         T1=X(I4)+X(I3)
         T2=X(I1)+X(I2)
C
C     X(I4) is Y(3N/4) of eqn. (4.7f):
C
         X(I4)=X(I3)-X(I4)
C
C     X(I2) is Y(N/4) of eqn. (4.7e):
C
         X(I2)=X(I1)-X(I2)
C
C     X(I1) is Y(0) of eqn. (4.7a):
C
         X(I1)=T2+T1
C
C     X(I3) is Y(N/2) of eqn. (4.7b):
C
         X(I3)=T2-T1
         I1=I1+N8
         I2=I2+N8
         I3=I3+N8
         I4=I4+N8
         T1=(X(I3)+X(I4))/SQRT(2.0)
         T2=(X(I3)-X(I4))/SQRT(2.0)
C
C     X(I4) is Y(7N/8) of eqn. (4.7h):
C
         X(I4)=X(I2)+T1
```

```
C
C     X(I3) is Y(5N/8) of eqn. (4.7g):
C
          X(I3)=-X(I2)+T1
C
C     X(I2) is Y(3N/8) of eqn. (4.7d):
C
          X(I2)=X(I1)-T2
C
C     X(I1) is Y(N/8) of eqn. (4.7c):
C
          X(I1)=X(I1)+T2
38        CONTINUE
C-----------------------------------------------------------------
        A=E
        DO 32 J=2,N8
            A2=2*A
            A3=3*A
            CC1=COS(A)
            SS1=SIN(A)
            CC2=COS(A2)
            SS2=SIN(A2)
            CC3=COS(A3)
            SS3=SIN(A3)
            A=J*E
            DO 30 I=0,N-1,N2
                I1=I+J
                I2=I1+N4
                I3=I2+N4
                I4=I3+N4
                I5=I+N4-J+2
                I6=I5+N4
                I7=I6+N4
                I8=I7+N4
C
C     T1 and T2  are a and b of eqn. (4.4):
C
                T1=(X(I3)*CC1-X(I7)*SS1)
                T2=(X(I7)*CC1+X(I3)*SS1)
C
```

```
C     T3 and T4 are e and f of eqn. (4.6):
C
                  T3=(X(I4)*CC3-X(I8)*SS3)
                  T4=(X(I8)*CC3+X(I4)*SS3)
C
C     P1 and P2 are c and d of eqn. (4.5):
C
                  P1=(X(I2)*CC2-X(I6)*SS2)
                  P2=(X(I6)*CC2+X(I2)*SS2)
                  T5=T1+T3
                  T3=T1-T3
                  T6=-T2-T4
                  T4=-T2+T4
                  T2=X(I5)+T6
C
C     X(I8) is Y(N-n) of eqn. (4.7p):
C
                  X(I8)=X(I5)-T6+P2
C
C     X(I3) is Y(N/2+n) of eqn. (4.7o):
C
                  X(I3)=-T2-P2
                  T2=X(I5)+T3
C
C     X(I4) is Y(3N/4+n) of eqn. (4.7l):
C
                  X(I4)=-X(I5)+T3+P2
C
C     X(I7) is Y(3N/4-n) of eqn. (4.7k):
C
                  X(I7)=T2-P2
                  T1=X(I1)-T4
C
C     X(I5) is Y(N/4-n) of eqn. (4.7m):
C
                  X(I5)=T1-P1
C
C     X(I2) is Y(N/4+n) of eqn. (4.7n):
C
                  X(I2)=X(I1)+T4-P1
```

```
                T1=X(I1)+T5
C
C     X(I6) is Y(N/2-n) of eqn. (4.7j):
C
                X(I6)=X(I1)-T5+P1
C
C     X(I1) is Y(n) of eqn. (4.7i):
C
                X(I1)=T1+P1
30          CONTINUE
32        CONTINUE
10      CONTINUE
      RETURN
      END
```

APPENDIX D

```
C
C      The split-radix decimation-in-time FRFT program
C      The output is in order:
C      [re(0),re(1),...,re(N/2),im(N/2-1),im(N/2-2),...,im(1)]
C
       SUBROUTINE FRFT(X,N)
       REAL X(1)
       M=INT(ALOG(FLOAT(N))/ALOG(2.0)+0.5)
C
C      Bit-reverse permutation of input data :
C
       J=1
       N1=N-1
       DO 104 I=1,N1
         IF(I.GE.J)GOTO 101
         XT=X(J)
         X(J)=X(I)
         X(I)=XT
101      K=N/2
102        IF(K.GE.J)GOTO 103
         J=J-K
         K=K/2
         GOTO 102
103      J=J+K
104    CONTINUE
C ---------------------------------------------------------------------
       IS=1
       ID=4
C
C      DO 60 loop below is size 2 R1:
C
70     DO 60 I0=IS,N,ID
         I1=I0+1
         R1=X(I0)
         X(I0)=R1+X(I1)
         X(I1)=R1-X(I1)
60     CONTINUE
       IS=2*ID-1
       ID=4*ID
```

```
      IF(IS.LT.N)GOTO 70
C -------------------------------------------------------------
        N2=2
        DO 10 K=2,M
C
C     Computation of size 2**k R1:
C
          N2=N2*2
          N4=N2/4
          N8=N2/8
          E=6.283185307179586/N2
          IS=0
          ID=N2*2
40       DO 38 I=IS,N-1,ID
          I1=I+1
          I2=I1+N4
          I3=I2+N4
          I4=I3+N4
          T1=X(I4)+X(I3)
C
C     X(I4) is Y(3N/4) of eqn. (5.2f):
C
          X(I4)=X(I3)-X(I4)
C
C     X(I3) is Y(N/2) of eqn. (5.2b):
C
          X(I3)=X(I1)-T1
C
C     X(I1) is Y(0) of eqn. (5.2a):
C
          X(I1)=X(I1)+T1
          IF(N4.EQ.1)GOTO 38
          I1=I1+N8
          I2=I2+N8
          I3=I3+N8
          I4=I4+N8
          T1=(X(I3)+X(I4))/SQRT(2.0)
          T2=(X(I3)-X(I4))/SQRT(2.0)
C
C     X(I4) is Y(7N/8) of eqn. (5.2h):
```

```
C
          X(I4)=X(I2)+T1
C
C     X(I3) is Y(5N/8) of eqn. (5.2g):
C
          X(I3)=-X(I2)+T1
C
C     X(I2) is Y(3N/8) of eqn. (5.2d):
C
          X(I2)=X(I1)-T2
C
C     X(I1) is Y(N/8) of eqn. (5.2c):
C
          X(I1)=X(I1)+T2
38        CONTINUE
          IS=2*ID-N2
          ID=4*ID
          IF(IS.LT.N) GOTO 40
          A=E
          DO 32 J=2,N8
             A3=3*A
             CC1=COS(A)
             SS1=SIN(A)
             CC3=COS(A3)
             SS3=SIN(A3)
             A=J*E
             IS=0
             ID=2*N2
36           DO 30 I=IS,N-1,ID
                I1=I+J
                I2=I1+N4
                I3=I2+N4
                I4=I3+N4
                I5=I+N4-J+2
                I6=I5+N4
                I7=I6+N4
                I8=I7+N4
C
C     The seven lines below compute  T1=(X(i3)*CC1-X(i7)*SS1) ,
C     T2=(X(i7)*CC1+X(i3)*SS1); T1 and T2 are a and b of eqn. (4.4):
```

```
C
                P1=CC1+SS1
                P2=SS1-CC1
                P3=CC1*(X(I3)+X(I7))
                P4=P1*X(I7)
                P5=P2*X(I3)
                T1=P3-P4
                T2=P3+P5
C
C     The seven lines below compute  T3=(X(i4)*CC3-X(i8)*SS3) ,
C     T4=(X(i8)*CC3+X(i4)*SS3); T3 and T4 are e and f of eqn. (4.6):
C
                P1=CC3+SS3
                P2=SS3-CC3
                P3=CC3*(X(I4)+X(I8))
                P4=P1*X(I8)
                P5=P2*X(I4)
                T3=P3-P4
                T4=P3+P5
                T5=T1+T3
                T6=T2+T4
                T3=T1-T3
                T4=T2-T4
                T2=X(I6)-T6
C
C     X(I8) is Y(N-n) of eqn. (5.2p):
C
                X(I8)=T6+X(I6)
C
C     X(I3) is Y(N/2+n) of eqn. (5.2o):
C
                X(I3)=-T2
                T2=X(I2)-T3
C
C     X(I4) is Y(3N/4-n) of eqn. (5.2k):
C
                X(I4)=X(I2)+T3
C
C     X(I7) is Y(3N/4+n) of eqn. (5.2l):
C
```

```
                X(I7)=-T2
                T1=X(I1)+T5
C
C     X(I6) is Y(N/2-n) of eqn. (5.2j):
C
                X(I6)=X(I1)-T5
C
C     X(I1) is Y(n) of eqn. (5.2i):
C
                X(I1)=T1
                T1=X(I5)-T4
C
C     X(I5) is Y(N/4-n) of eqn. (5.2m):
C
                X(I5)=X(I5)+T4
C
C     X(I2) is Y(N/4+n) of eqn. (5.2n):
C
                X(I2)=T1
30          CONTINUE
            IS=2*ID-N2
            ID=4*ID
            IF(IS.LT.N)GOTO 36
32        CONTINUE
10      CONTINUE
      RETURN
      END
```

APPENDIX E

```
C
C      The split-radix decimation-in-frequency FRFT program
C      The output is in order:
C      [re(0),re(1),...re(N/2),im(N/2-1),im(N/2-2),...,im(1)]
C
       SUBROUTINE FRFT(X,N,FX)
C
C      x=input data ,fx=output data
       REAL X(1), FX(1)
       INTEGER N
C
       INTEGER NSQ(100), NOL(51)
       INTEGER INP, NN, L, NK, I, J, LP, LGN
C
C      Take care of initial A (if necessary), compute blocksize (NN),
C     compute number of stage pairs (L)
C
       LGN = INT(ALOG(FLOAT(N))/ALOG(2.0) + 0.5)
       IF (MOD(LGN,2) .NE. 0) THEN
         CALL ADD(X,N)
         NN=N/2
C      L below is the number of stage pairs consisting of two
C      successive stages :
         L=(LGN-1)/2
       ELSE
         NN=N
         L=LGN/2
       END IF
C
C     Generate control sequence for this value of N
C     (NSQ is the parameter control sequence SCN(p. 17);
C      NK is the number of IAS operations of size 8;
C      NOL is an auxiliary sequence used to step through NSQ)
C
       CALL BASQ(NSQ,NK,LGN,NOL)
C
C      Process the data
C
```

```
      DO 1000 LP=1,L
        INP=0
        J = 1
        DO 1100 I=1,N/NN
          IF (I .NE. (NSQ(J)+1)) THEN
            CALL ADDPLANE(X(1+INP),NN)
          ELSE
            CALL PLANEADD(X(1+INP),NN)
            J = J + 1
          ENDIF
          INP=INP+NN
1100    CONTINUE
        NN=NN/4
1000  CONTINUE
C
C      Perform IAS to combine results of small R1's
C      (LP below is the number of block sizes from 8 to N/2;
C       NK is the number of IAS operations of size J)
C
      NN = 8
      LP = LGN-3
      DO 2000 J = 1,LP
        DO 2100 I = 1,NK
          INP = NN * NSQ(I)
          CALL ADDSUB(X,N,NN,INP)
2100    CONTINUE
        NN = 2*NN
        NK = NK-NOL(LP-J+1)
2000  CONTINUE
C
C      Permute output results
C
      CALL LKUP(X,N,L,FX)
C
      RETURN
      END
C
C
      SUBROUTINE ADDSUB(X,N,M,INP)
C
```

```
C     Subroutine ADDSUB performs an IAS of size M
C
      REAL X(1)
      INTEGER N, M, INP
C
      INTEGER START, END, M2, L, I, J, NS, NE, LGM8
      REAL XT
C
C     Define and adjust initial block
C
      M2 = M/2
      START = INP + 3
      END = INP + M2 + 4
      IF ((START .GT. N) .OR. (END .GT. N)) GO TO 10
      XT = X(START)
      X(START) = XT + X(END)
      X(END) = XT - X(END)
10    IF (((START+1) .GT. N) .OR. ((END-1) .GT. N)) GO TO 20
      XT = X(START+1)
      X(START+1) = XT + X(END-1)
      X(END-1) = XT - X(END-1)
20    IF (M .EQ. 8) GO TO 1
C
      LGM8 = INT(ALOG(M/8.0)/ALOG(2.0) + 0.5)
      NS = START + 1
      L = 1
      DO 100 I = 1,LGM8
        L = 2 * L
C
C     Adjust X1 (6.9)
C
        DO 110 J = 1,L
          NS = NS + 1
          NE = NS + M2 + L
          IF ((NS .GT. N) .OR. (NE .GT. N)) GO TO 110
          XT = X(NS)
          X(NS) = XT + X(NE)
          X(NE) = XT - X(NE)
110     CONTINUE
C
```

```
C     Adjust X0 (eqn. 6.10)
C
        DO 120 J = 1,L
          NS = NS + 1
          NE = NS + M2 - L
          IF ((NS .GT. N) .OR. (NE .GT. N)) GO TO 120
          XT = X(NS)
          X(NS) = XT + X(NE)
          X(NE) = XT - X(NE)
120     CONTINUE
100   CONTINUE
C
1     CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE ADDPLANE(X,M)
C
C      Subroutine ADDPLANE implements an AAP of size M
C
      REAL X(1)
      INTEGER M
C
      INTEGER M2
C
      M2 = M/2
      CALL ADD(X,M)
      CALL ADD(X,M2)
      CALL PLANE(X(1+M2),M2)
C
      RETURN
      END
C
C
C
      SUBROUTINE PLANEADD(X,M)
C
C      Subroutine PLANEADD implements a PAA of size M
```

```
C
      REAL X(1)
      INTEGER M
C
      INTEGER M2
C
      M2 = M/2
      CALL PLANE(X,M)
      CALL ADD(X,M2)
      CALL ADD(X(1+M2),M2)
C
      RETURN
      END
C
C
C
      SUBROUTINE ADD(X,M)
C
C     Subroutine ADD implements an A of size M(eqns. 3.1a and 6.6)
C
      REAL X(1)
      INTEGER M
C
      REAL XT
      INTEGER I, M2, M2PI
C
      M2=M/2
      DO 100 I=1,M2
        M2PI = M2 + I
        XT = X(I)
        X(I) = XT + X(M2PI)
        X(M2PI) = XT - X(M2PI)
100   CONTINUE
C
      RETURN
      END
C
      SUBROUTINE PLANE(X,M)
C
C     Subroutine PLANE implements a P of size M
```

```
C
      REAL X(1)
      INTEGER M
C
      INTEGER I, J, M2, M3
      REAL PI2, ANG, HC, HS, R0, R1
C
      PI2 = 8.0*ATAN(1.0)
      M2 = M/2
      M3 = 2*M
      DO 200 I = 1,M2
        ANG = PI2*(I-1)/(FLOAT(M3))
        HC = COS(ANG)
        HS = SIN(ANG)
        J = I + M2
        R0 = HC * (X(I) + X(J))
        R1 = (HS - HC) * X(I)
        X(I) = R0 - (HC + HS) * X(J)
        X(J) = - (R0 + R1)
200   CONTINUE
C
      RETURN
      END
C
C
C
      SUBROUTINE BASQ(NSQ,K,LGN,NOL)
C
C     Subroutine BASQ generates the control sequence parameters
C
      INTEGER NSQ(1), K, LGN, NOL(1)
C
      INTEGER L, I, NN, M, J
C
C     Define the basic sequences...
C
      NOL(1)=-1
      NOL(2)=1
      NOL(3)=2
      NOL(4)=2
```

```
      NSQ(1)=1
      NSQ(2)=5
      NSQ(3)=7
      NSQ(4)=9
      NSQ(5)=13
C
      L=LGN-4
      DO 100 I=4,L+1
        NOL(I+1)=NOL(I)+2*NOL(I-1)
100   CONTINUE
C
      K=1
      NN=16
      DO 200 J=1,L+1
        NN=2*NN
        M=NOL(J+1)/2
        DO 300 I=1,M
          NSQ(I+K)=NSQ(I)+NN/16
          NSQ(I+K+M)=NSQ(I)+3*NN/32
300     CONTINUE
        K = 2*M + K
200   CONTINUE
C
      RETURN
      END
C
C
C
      SUBROUTINE LKUP(X,N,L,FX)
C
C     Subroutine LKUP permutes the output data and changes
C     the sign of appropriate output values (eqn. 6.4)
C
      REAL X(1), FX(1)
      INTEGER N, L
C
      INTEGER NSEQ(256), I, K, ND, NS, M, NQ
C
      OPEN(UNIT=2,FILE='LOOKUP')
      REWIND(2)
```

```
      READ(2,*) (NSEQ(I),I=1,256)
C
      K = 0
      ND = 1
      DO 100 I=1,L
        ND = ND * 4
        K = K + N/ND
100   CONTINUE
      NS = N - K
      M = 256/N
      DO 200 I=1,N
        NQ = NSEQ(I)/M
        IF(NQ.GE.NS)THEN
          FX(NQ+1) = -X(I)
        ELSE
          FX(NQ+1) = X(I)
        END IF
200   CONTINUE
C
      RETURN
      END
```

REFERENCES

1. O. K. Ersoy, "Real Discrete Fourier Transform," *IEEE Tran. Acoustics, Speech, Signal Processing,* Vol. ASSP-33, No. 4, 880-882, August 1985.
2. N. Ahmed, T. Natarajan, K. R. Rao, "Discrete Cosine Transform," *IEEE Tran. Computers,* Vol. C-23, 90-93, January 1974.
3. A. K. Jain, "A Fast Karhunen-Loeve Transform for a Class of Stochastic Processes," *IEEE Tran. Commun,* Vol. COM-24, 1023-1029, 1976.
4. R. N. Bracewell, "The Discrete Hartley Transform," *J. Optical Society of America,* Vol. 73, 1832-1835, December 1983.
5. O. K. Ersoy, M. Zeng, "New Approaches to Generalized Matched Filtering," *Optical Society of America Annual Meeting,* Rochester, New York, October, 1987, and submitted to *J. Optical Society of America, A.*
6. O. K. Ersoy, "On Walsh-Domain Versus RDFT Filtering," *IEEE Tran. Acoustics, Speech, Signal Processing,* in print.
7. O. K. Ersoy, N. C. Hu, "A Unified Approach to the Fast Computation of All Discrete Trigonometric Transforms," *ICASSP 1987 Proceedings,* 1843-1847, Dallas, April 1987.
8. O. K. Ersoy, C-H Chen, "Transform-Coding of Images with Reduced Complexity," *Computer Vision, Graphics, and Image Processing,* in print.
9. O. K. Ersoy, "Generalized Real-Discrete Fourier Transform: A Family of Transforms," *Conf. Information Sciences and Systems Proceedings,* 777-782, Princeton University, March, 1986
10. H. V. Sorensen, D. L. Jones, M. T. Heideman, C. S. Burrus, "Real-Valued Fast Fourier Transform Algorithms," *IEEE Tran. Acoustics, Speech, Signal Processing,* Vol. ASSP-35, No. 6, June 1987.
11. P. Duhamel, H. Hollman, "Split-Radix FFT Algorithm," *Electron Lett.,* Vol. 20, 14-16, January 1984.
12. I. J. Good, "The Interaction Algorithm and Practical Fourier Analysis," *J. Royal Statist. Soc., Ser. V,* Vol. 20, 361-375, 1958, Vol. 22, 372-375, 1960.
13. L. H. Thomas, "Using a Computer to Solve Problems in Physics," *Applications of Digital Computers,* Ginn and Co., Boston, Mass., 1963.
14. R. E. Blahut, *Fast Algorithms for Digital Signal Processing,* Addison-Wesley, 1985.
15. O. K. Ersoy, "A Real-Time Interpolation of Images Obtained by Image Detector Arrays," *Conf. Image Detection and Quality,* 371-374, Paris, July, 1986.
16. C. M. Rader, "Discrete Fourier Transform When the Number of Data Samples is Prime," *Proc. IEEE,* Vol. 56, 1107-1108, 1968.
17. "A Real Formalism of Discrete Fourier Transform in Terms of Skew-Circular Correlations" and its Computation by Fast Correlation Techniques, *SPIE Conf. Real Time Signal Processing VI,* 239-253, San Diego, September 1983.
18. S. Winograd, "On Computing the Discrete Fourier Transform," *Math. Computation,* Vol. 32, No. 141, 175-199, January 1978.

Table 1. The Output Permutation Sequence in FRFT for $N \leqq 256$.

0,128,64,192,32,96,224,160,16,112,48,176,240,144,208,80,
8,120,56,184,24,104,216,168,248,136,200,72,232,152,40,88,
4,124,60,188,28,100,220,164,12,116,44,172,236,148,204,76,
252,132,196,68,228,156,36,92,244,140,212,84,20,108,52,180,
2,126,62,190,30,98,222,162,14,114,46,174,238,146,206,78,
6,122,54,182,22,106,214,170,246,138,198,70,230,154,38,90,
254,130,194,66,226,158,34,94,242,142,210,82,18,110,50,178,
250,134,202,74,234,150,42,86,10,118,58,186,26,102,218,166,
1,127,63,191,31,97,223,161,15,113,47,175,239,145,207,79,
7,121,55,183,23,105,215,169,247,137,199,71,231,153,39,89,
3,125,59,187,27,101,219,165,11,117,43,171,235,149,203,75,
251,133,195,67,227,157,35,93,243,141,211,83,19,109,51,179,
255,129,183,65,225,159,33,95,241,143,209,81,17,111,49,177,
249,135,201,73,233,151,41,87,9,119,57,185,25,103,217,167,
253,131,197,69,229,155,37,91,245,139,213,85,21,107,53,181,
5,123,61,189,29,99,221,163,13,115,45,173,237,147,205,77

Table 2. The Relationship Between the 2-D DFT and the 2-D RDFT.

n_1	n_2	$X_1(n_1,n_2)$	$X_1(N_1-n_1,n_2)$	$X_0(n_1,n_2)$	$X_0(N_1-n_1,n_2)$
0	0	$y(0,0)$	$y(0,)$	0	0
$N_1/2$	$N_2/2$	$y(n_1,n_2)$	$y(n_1,n_2)$	0	0
0	$0<n_2<N_2/2$	$y(0,n_2)$	$y(0,n_2)$	$y(0,N_2-n_2)$	$y(0,N_2-n_2)$
$N_1/2$	$0<n_2<N_2/2$	$y(N_1/2,n_2)$	$y(N_1/2,n_2)$	$y(N_1/2,N_2-n_2)$	$y(N_1/2,N_2-n_2)$
$0<n_1<N_1/2$	0	$y(n_1,0)$	$y(n_1,0)$	$y(N_1-n_1,0)$	$-y(N_1-n_1,0)$
$0<n_1<N_1/2$	$N_2/2$	$y(N_1,N_2/2)$	$y(n_1,N_2/2)$	$y(N_1-n_1,N_2/2)$	$-y(N_1-n_1,N_2/2)$
$0<n_1<N_1/2$	$0<n_2<N_2/2$	$y(n_1,n_2)-y(N_1-n_1,N_2-n_2)$	$y(n_1,n_2)+y(N_1-n_1,N_2-n_2)$	$y(n_1,N_2-n_2)+y(N_1-n_1,n_2)$	$-y(n_1N_2-n_2)+y(N_1-n_1,n_2)$

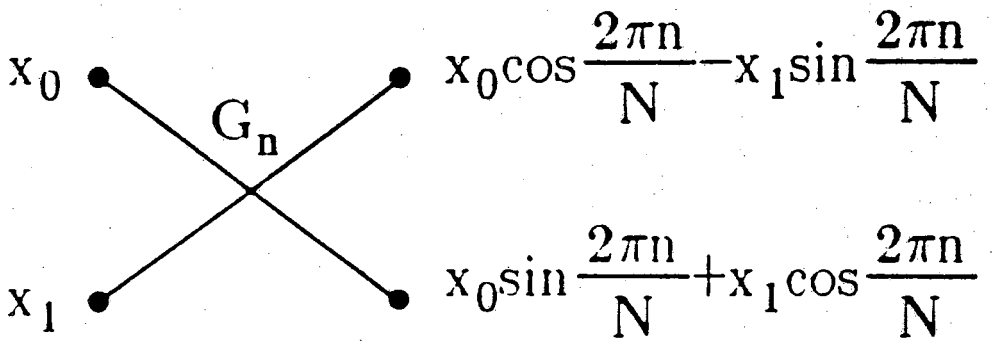


Fig. 1. Givens' Plane Rotation, the Basic Operation in the FRFT Algorithms.

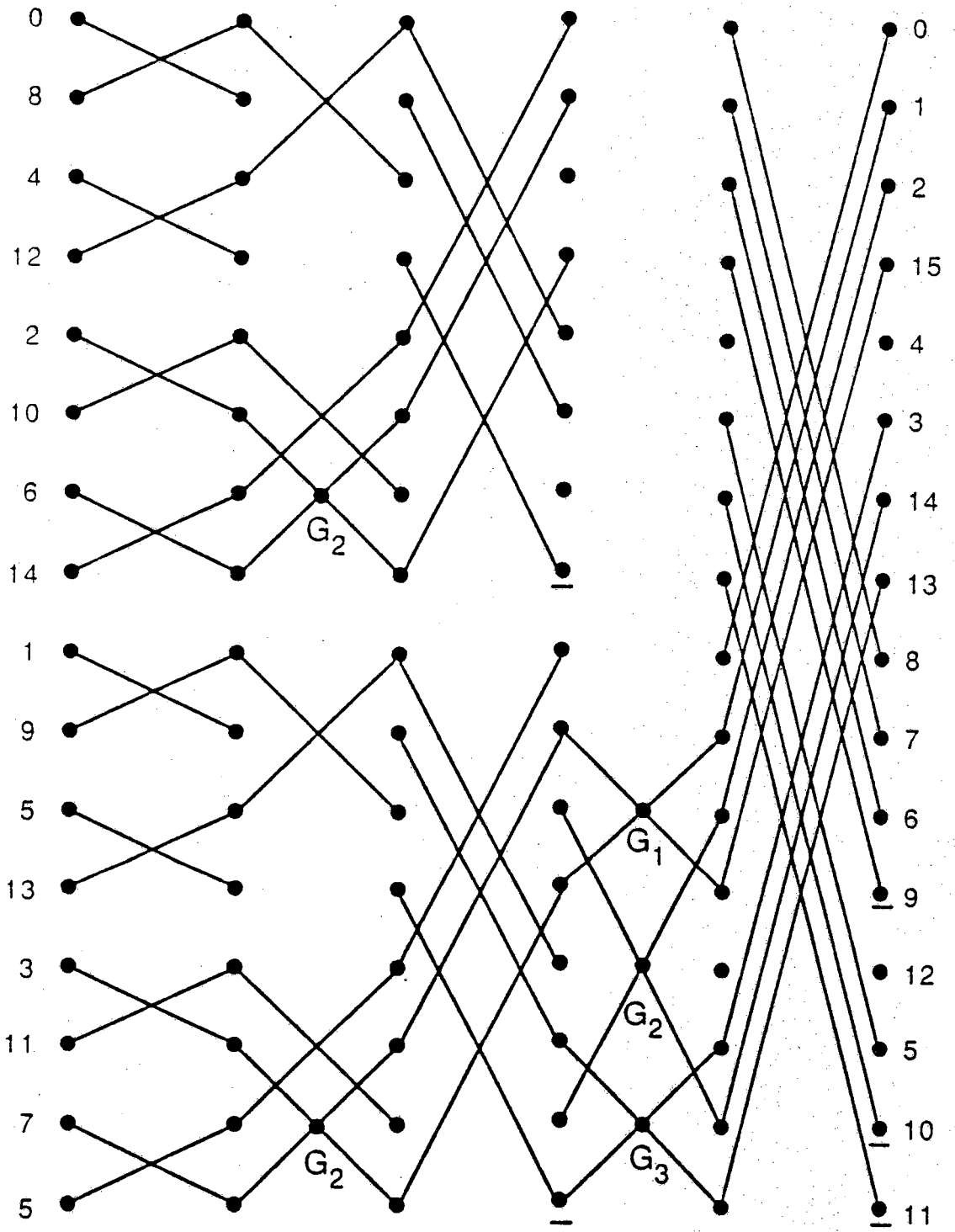


Fig. 2. The DIT FRFT Signal-Flow Diagram When N equals 16.

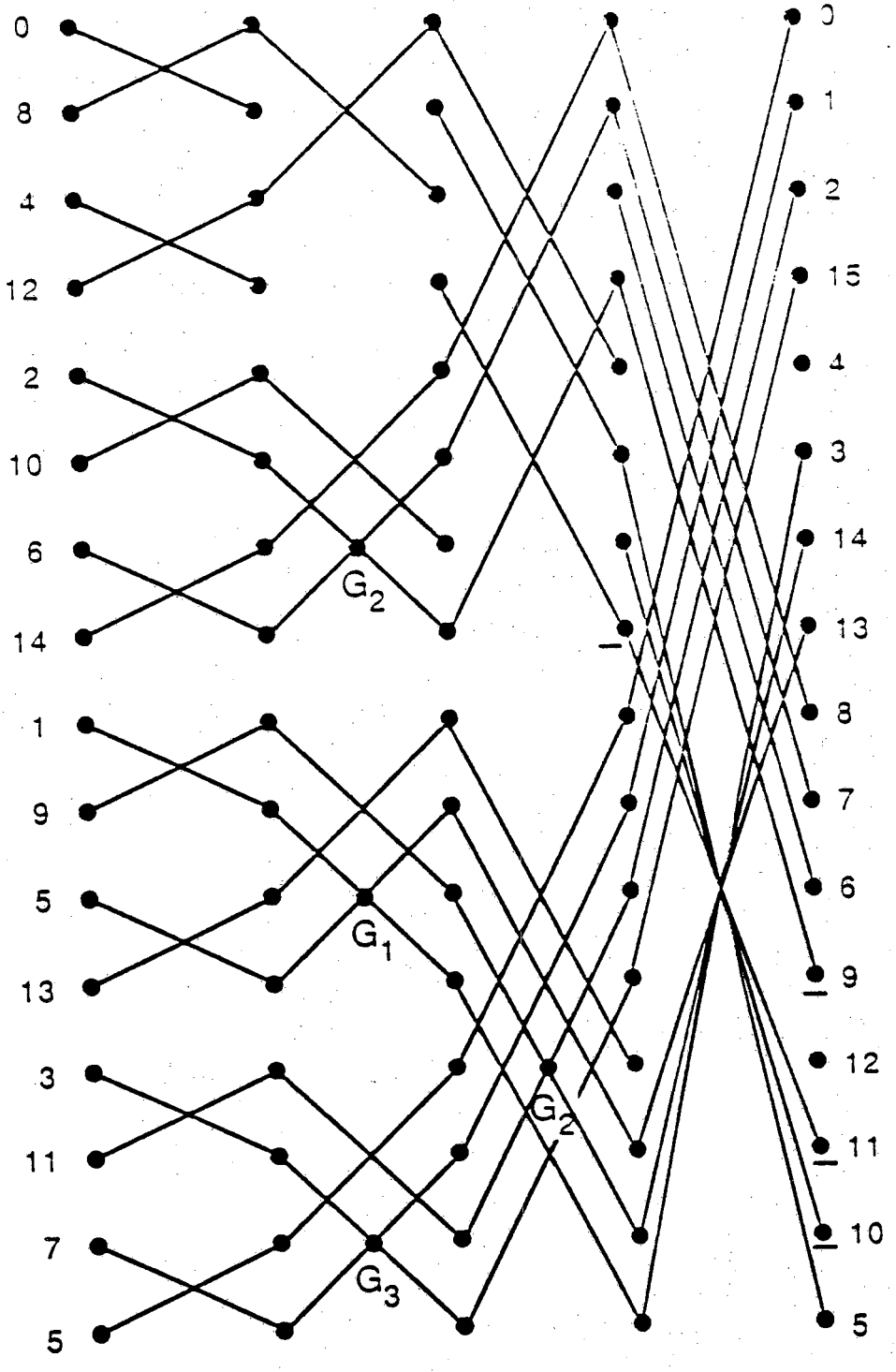


Fig. 3. The Split-Radix DIT FRFT Signal-Flow Graph When N=16.

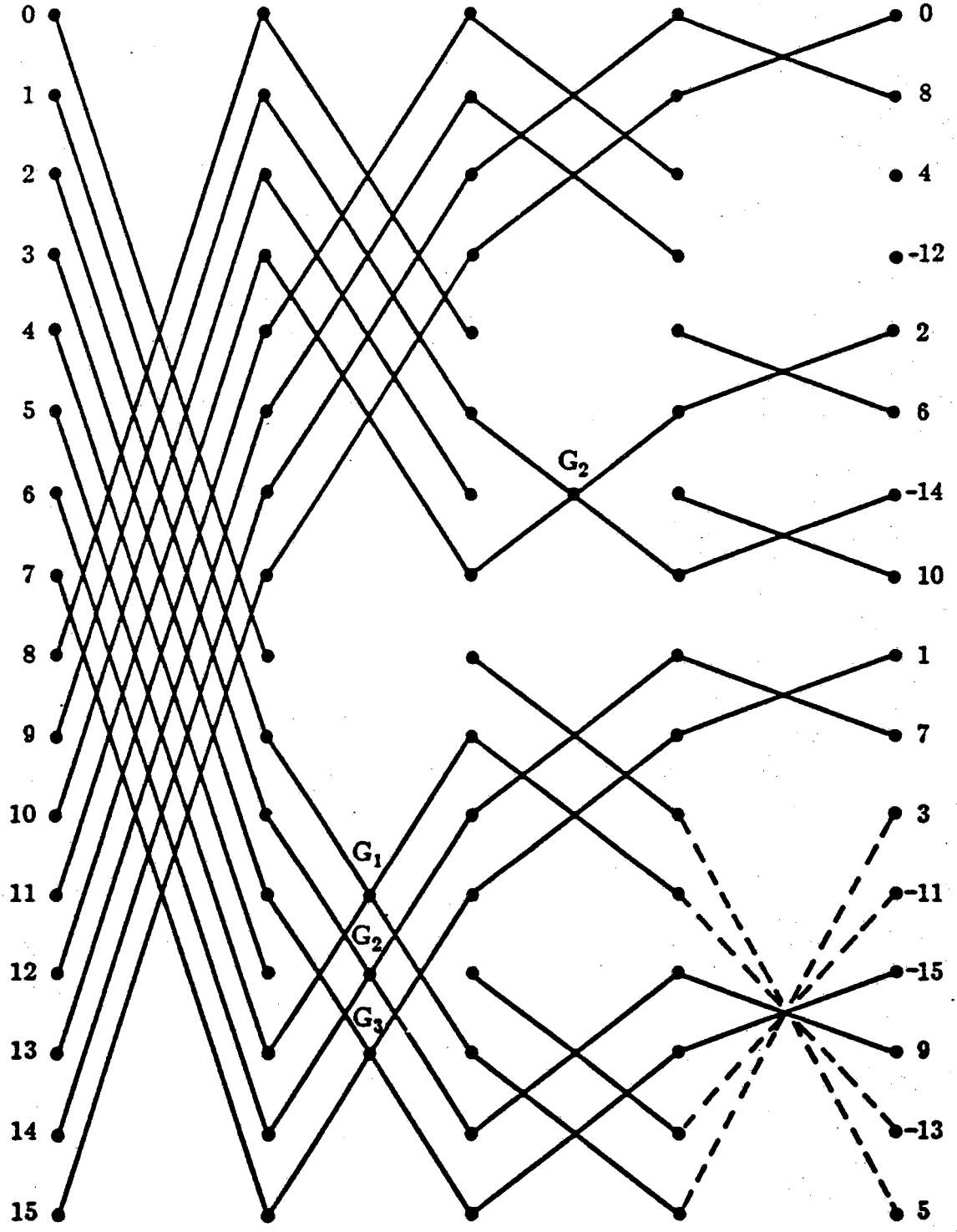


Fig. 4. The Split-Radix DIF FRFT Signal-Flow Graph When N=16.

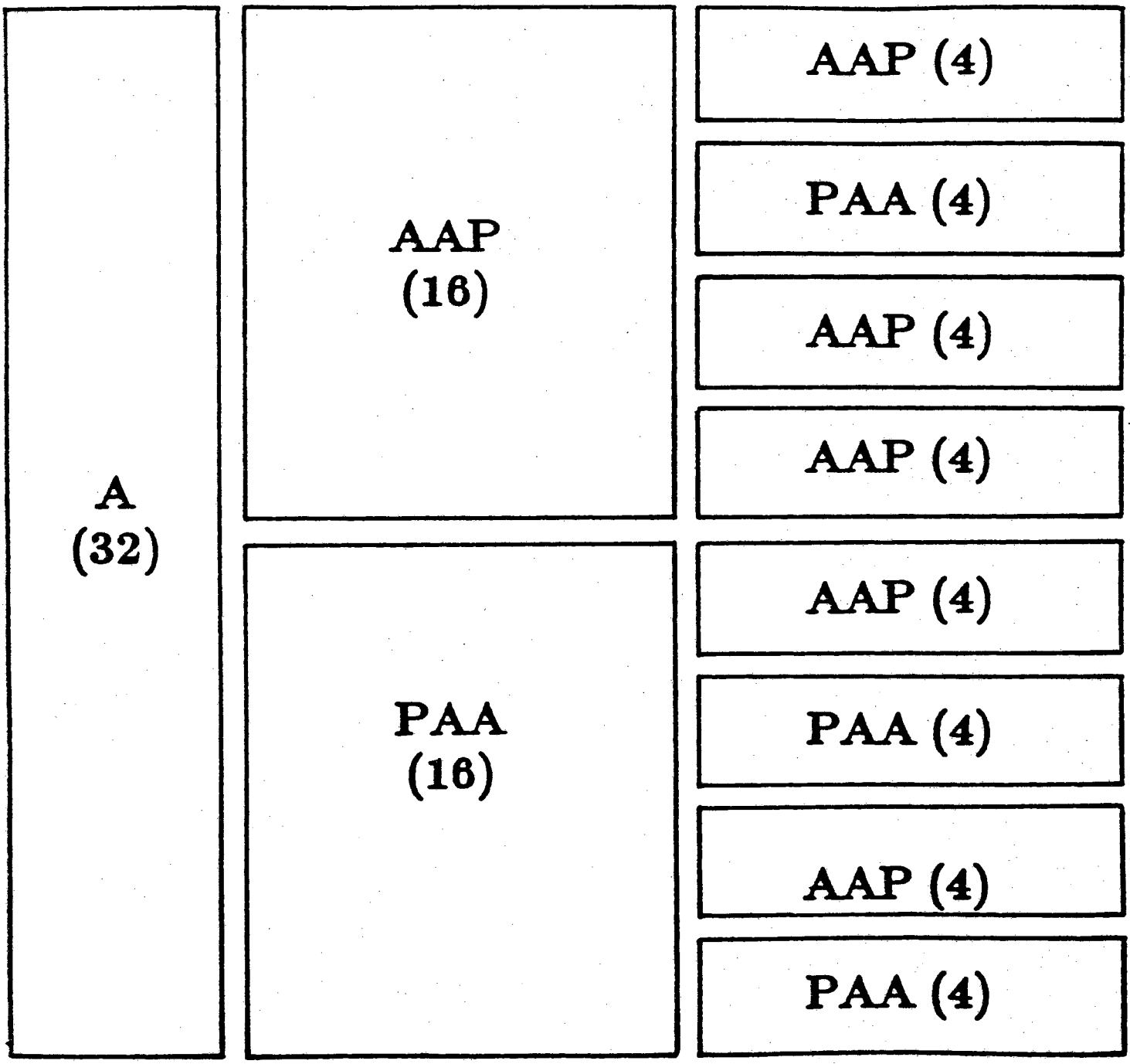


Fig. 5. The Block Diagram of A, AAP and PAA Operations When N=32.

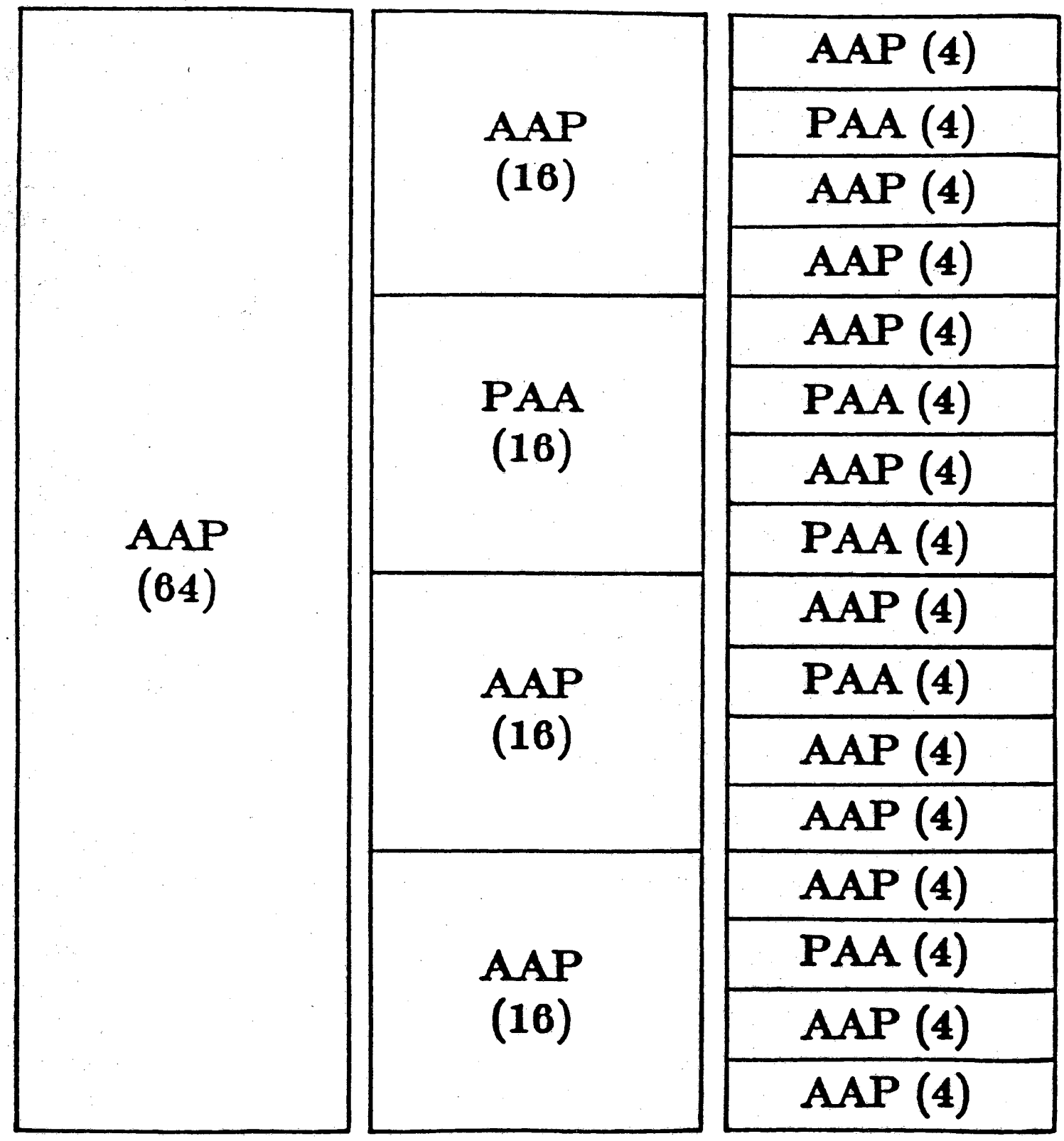


Fig. 6. The Block Diagram of AAP and PAA Operations When N=64.