9 COMPUTATION OF THE DISCRETE FOURIER TRANSFORM

9.1 EFFICIENT COMPUTATION OF THE DISCRETE FOURIER TRANSFORM

As defined in Chapter 8, the DFT of a finite-length sequence of length N is

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{kn}, \qquad k = 0, 1, \ldots, N-1, \tag{9.1}$$

where $W_N = e^{-j(2\pi/N)}$. The inverse discrete Fourier transform is given by

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]W_N^{-kn}, \qquad n = 0, 1, \ldots, N-1. \tag{9.2}$$

In Eqs. (9.1) and (9.2), both $x[n]$ and $X[k]$ may be complex. Since the expressions on the right-hand sides of those equations differ only in the sign of the exponent of W_N and in the scale factor $1/N$, a discussion of computational procedures for Eq. (9.1) applies with straightforward modifications to Eq. (9.2). (See Problem 9.1.)

To create a frame of reference for our discussion of computation of the DFT, let us first consider direct evaluation of the DFT expression in Eq. (9.1). Since $x[n]$ may be complex, N complex multiplications and $(N-1)$ complex additions are required to compute each value of the DFT if we use Eq. (9.1) directly as a formula for computation. To compute all N values therefore requires a total of N^2 complex multiplications and $N(N-1)$ complex additions.

Most approaches to improving the efficiency of the computation of the DFT exploit the symmetry and periodicity properties of W_N^{kn}; specifically,

1. $W_N^{k[N-n]} = W_N^{-kn} = (W_N^{kn})^*$ (complex conjugate symmetry);
2. $W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n}$ (periodicity in n and k).

9.3 DECIMATION-IN-TIME FFT ALGORITHMS

In computing the DFT, dramatic efficiency results from decomposing the computation into successively smaller DFT computations. In this process, we exploit both the symmetry and the periodicity of the complex exponential $W_N^{kn} = e^{-j(2\pi/N)kn}$. Algorithms in which the decomposition is based on decomposing the sequence $x[n]$ into successively smaller subsequences are called *decimation-in-time algorithms*.

The principle of the decimation-in-time algorithm is most conveniently illustrated by considering the special case of N an integer power of 2, i.e., $N = 2^\nu$. Since N is an even integer, we can consider computing $X[k]$ by separating $x[n]$ into two $(N/2)$-point[3] sequences consisting of the even-numbered points in $x[n]$ and the odd-numbered points in $x[n]$. With $X[k]$ given by

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{nk}, \qquad k = 0, 1, \ldots, N-1, \tag{9.10}$$

and separating $x[n]$ into its even- and odd-numbered points, we obtain

$$X[k] = \sum_{n \text{ even}} x[n]W_N^{nk} + \sum_{n \text{ odd}} x[n]W_N^{nk}, \tag{9.11}$$

or, with the substitution of variables $n = 2r$ for n even and $n = 2r+1$ for n odd,

$$\begin{aligned} X[k] &= \sum_{r=0}^{(N/2)-1} x[2r]W_N^{2rk} + \sum_{r=0}^{(N/2)-1} x[2r+1]W_N^{(2r+1)k} \\ &= \sum_{r=0}^{(N/2)-1} x[2r](W_N^2)^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1](W_N^2)^{rk}. \end{aligned} \tag{9.12}$$

But $W_N^2 = W_{N/2}$, since

$$W_N^2 = e^{-2j(2\pi/N)} = e^{-j2\pi/(N/2)} = W_{N/2}. \tag{9.13}$$

Consequently, Eq. (9.12) can be rewritten as

$$\begin{aligned} X[k] &= \sum_{r=0}^{(N/2)-1} x[2r]W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1]W_{N/2}^{rk} \\ &= G[k] + W_N^k H[k], \qquad k = 0, 1, \ldots, N-1. \end{aligned} \tag{9.14}$$

Each of the sums in Eq. (9.14) is recognized as an $(N/2)$-point DFT, the first sum being the $(N/2)$-point DFT of the even-numbered points of the original sequence and the second being the $(N/2)$-point DFT of the odd-numbered points of the original sequence. Although the index k ranges over N values, $k = 0, 1, \ldots, N-1$, each of the sums must be computed only for k between 0 and $(N/2)-1$, since $G[k]$ and $H[k]$ are each periodic in k with period $N/2$. After the two DFTs are computed, they are combined according to Eq. (9.14) to yield the N-point DFT $X[k]$. Figure 9.3 depicts this computation for $N = 8$. In this figure, we have used the signal flow graph conventions that were introduced in Chapter 6 for representing difference equations. That is, branches entering a node are summed to produce the node variable. When no coefficient is indicated, the branch transmittance is assumed to be unity. For other branches, the transmittance of a branch is an integer power of W_N.

In Figure 9.3, two 4-point DFTs are computed, with $G[k]$ designating the 4-point DFT of the even-numbered points and $H[k]$ designating the 4-point DFT of the odd-numbered points. $X[0]$ is then obtained by multiplying $H[0]$ by W_N^0 and adding the product to $G[0]$. $X[1]$ is obtained by multiplying $H[1]$ by W_N^1 and adding that result to $G[1]$. Equation (9.14) states that, to compute $X[4]$, we should multiply $H[4]$ by W_N^4 and add the result of $G[4]$. However, since $G[k]$ and $H[k]$ are both periodic in k with

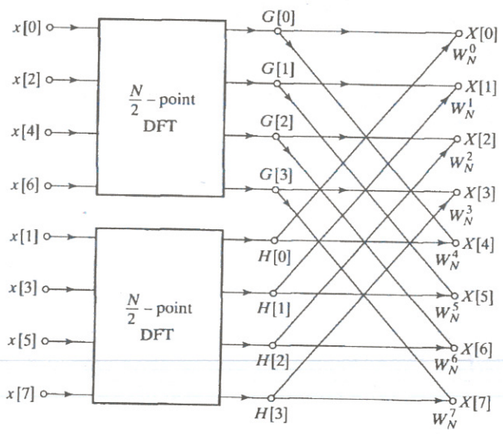

Figure 9.3 Flow graph of the decimation-in-time decomposition of an N-point DFT computation into two $(N/2)$-point DFT computations $(N = 8)$.

period 4, $H[4] = H[0]$ and $G[4] = G[0]$. Thus, $X[4]$ is obtained by multiplying $H[0]$ by W_N^4 and adding the result to $G[0]$. As shown in Figure 9.3, the values $X[5]$, $X[6]$, and $X[7]$ are obtained similarly.

With the computation restructured according to Eq. (9.14), we can compare the number of multiplications and additions required with those required for a direct computation of the DFT. Previously we saw that, for direct computation without exploiting symmetry, N^2 complex multiplications and additions were required.[4] By comparison, Eq. (9.14) requires the computation of two $(N/2)$-point DFTs, which in turn requires $2(N/2)^2$ complex multiplications and approximately $2(N/2)^2$ complex additions if we do the $(N/2)$-point DFTs by the direct method. Then the two $(N/2)$-point DFTs must be combined, requiring N complex multiplications, corresponding to multiplying the second sum by W_N^k, and N complex additions, corresponding to adding the product obtained to the first sum. Consequently, the computation of Eq. (9.14) for all values of k requires at most $N + 2(N/2)^2$ or $N + N^2/2$ complex multiplications and complex additions. It is easy to verify that for $N > 2$, the total $N + N^2/2$ will be less than N^2.

Equation (9.14) corresponds to breaking the original N-point computation into two $(N/2)$-point DFT computations. If $N/2$ is even, as it is when N is equal to a power of 2, then we can consider computing each of the $(N/2)$-point DFTs in Eq. (9.14) by breaking each of the sums in that equation into two $(N/4)$-point DFTs, which would then be combined to yield the $(N/2)$-point DFTs. Thus, $G[k]$ in Eq. (9.14) would be represented as

$$G[k] = \sum_{r=0}^{(N/2)-1} g[r]W_{N/2}^{rk} = \sum_{\ell=0}^{(N/4)-1} g[2\ell]W_{N/2}^{2\ell k} + \sum_{\ell=0}^{(N/4)-1} g[2\ell+1]W_{N/2}^{(2\ell+1)k}, \tag{9.15}$$

or

$$G[k] = \sum_{\ell=0}^{(N/4)-1} g[2\ell]W_{N/4}^{\ell k} + W_{N/2}^k \sum_{\ell=0}^{(N/4)-1} g[2\ell+1]W_{N/4}^{\ell k}. \tag{9.16}$$

Similarly, $H[k]$ would be represented as

$$H[k] = \sum_{\ell=0}^{(N/4)-1} h[2\ell]W_{N/4}^{\ell k} + W_{N/2}^k \sum_{\ell=0}^{(N/4)-1} h[2\ell+1]W_{N/4}^{\ell k}. \tag{9.17}$$

Consequently, the $(N/2)$-point DFT $G[k]$ can be obtained by combining the $(N/4)$-point DFTs of the sequences $g[2\ell]$ and $g[2\ell+1]$. Similarly, the $(N/2)$-point DFT $H[k]$ can be obtained by combining the $(N/4)$-point DFTs of the sequences $h[2\ell]$ and $h[2\ell+1]$. Thus, if the 4-point DFTs in Figure 9.3 are computed according to Eqs. (9.16) and (9.17), then that computation would be carried out as indicated in Figure 9.4. Inserting the computation of Figure 9.4 into the flow graph of Figure 9.3, we obtain the complete flow graph of Figure 9.5, where we have expressed the coefficients in terms of powers of W_N rather than powers of $W_{N/2}$, using the fact that $W_{N/2} = W_N^2$.

For the 8-point DFT that we have been using as an illustration, the computation has been reduced to a computation of 2-point DFTs. For example, the 2-point DFT of the sequence consisting of $x[0]$ and $x[4]$ is depicted in Figure 9.6. With the computation

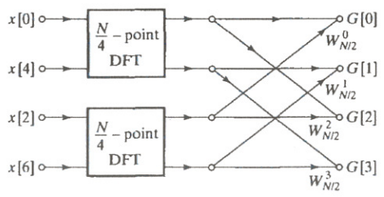

Figure 9.4 Flow graph of the decimation-in-time decomposition of an $(N/2)$-point DFT computation into two $(N/4)$-point DFT computations $(N = 8)$.

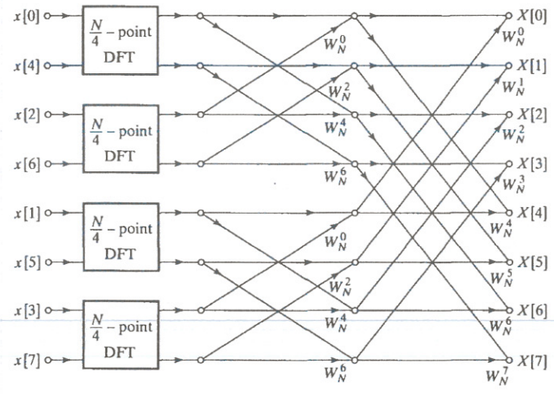

Figure 9.5 Result of substituting the structure of Figure 9.4 into Figure 9.3.

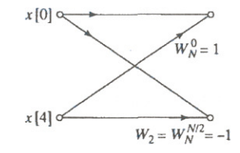

Figure 9.6 Flow graph of a 2-point DFT.

of Figure 9.6 inserted in the flow graph of Figure 9.5, we obtain the complete flow graph for computation of the 8-point DFT, as shown in Figure 9.7.

For the more general case, but with N still a power of 2, we would proceed by decomposing the $(N/4)$-point transforms in Eqs. (9.16) and (9.17) into $(N/8)$-point transforms and continue until we were left with only 2-point transforms. This requires $\nu = \log_2 N$ stages of computation. Previously, we found that in the original decomposition of an N-point transform into two $(N/2)$-point transforms, the number of complex multiplications and additions required was $N + 2(N/2)^2$. When the $(N/2)$-point transforms are decomposed into $(N/4)$-point transforms, the factor of $(N/2)^2$ is replaced by $N/2 + 2(N/4)^2$, so the overall computation then requires $N + N + 4(N/4)^2$ complex multiplications and additions. If $N = 2^\nu$, this can be done at most $\nu = \log_2 N$ times, so that after carrying out this decomposition as many times as possible, the number of complex multiplications and additions is equal to $N\nu = N\log_2 N$.

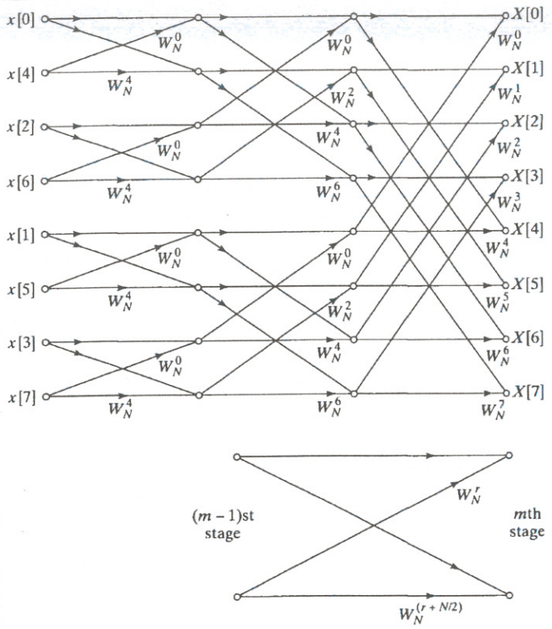

Figure 9.7 Flow graph of complete decimation-in-time decomposition of an 8-point DFT computation.

Figure 9.8 Flow graph of basic butterfly computation in Figure 9.7.

The flow graph of Figure 9.7 displays the operations explicitly. By counting branches with transmittances of the form W_N^r, we note that each stage has N complex multiplications and N complex additions. Since there are $\log_2 N$ stages, we have a total of $N\log_2 N$ complex multiplications and additions. This is the substantial computational savings that we have previously indicated was possible. For example, if $N = 2^{10} = 1024$, then $N^2 = 2^{20} = 1{,}048{,}576$, and $N\log_2 N = 10{,}240$, a reduction of more than two orders of magnitude!

The computation in the flow graph of Figure 9.7 can be reduced further by exploiting the symmetry and periodicity of the coefficients W_N^r. We first note that, in proceeding from one stage to the next in Figure 9.7, the basic computation is in the form of Figure 9.8, i.e., it involves obtaining a pair of values in one stage from a pair of values in the preceding stage, where the coefficients are always powers of W_N and the exponents are separated by $N/2$. Because of the shape of the flow graph, this elementary computation is called a *butterfly*. Since

$$W_N^{N/2} = e^{-j(2\pi/N)N/2} = e^{-j\pi} = -1, \tag{9.18}$$

the factor $W_N^{r+N/2}$ can be written as

$$W_N^{r+N/2} = W_N^{N/2} W_N^r = -W_N^r. \tag{9.19}$$

With this observation, the butterfly computation of Figure 9.8 can be simplified to the form shown in Figure 9.9, which requires only one complex multiplication instead of two. Using the basic flow graph of Figure 9.9 as a replacement for butterflies of the form of Figure 9.8, we obtain from Figure 9.7 the flow graph of Figure 9.10. In particular, the number of complex multiplications has been reduced by a factor of 2 over the number in Figure 9.7.

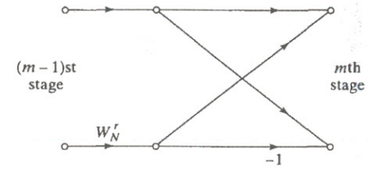

Figure 9.9 Flow graph of simplified butterfly computation requiring only one complex multiplication.

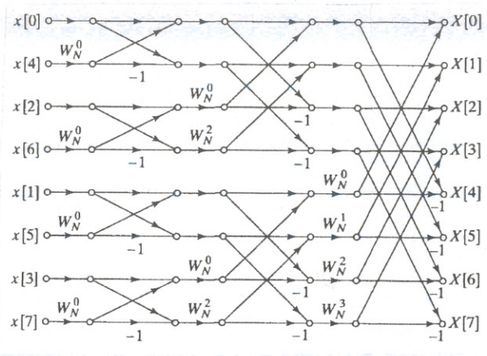

Figure 9.10 Flow graph of 8-point DFT using the butterfly computation of Figure 9.9.

9.3.2 Alternative Forms

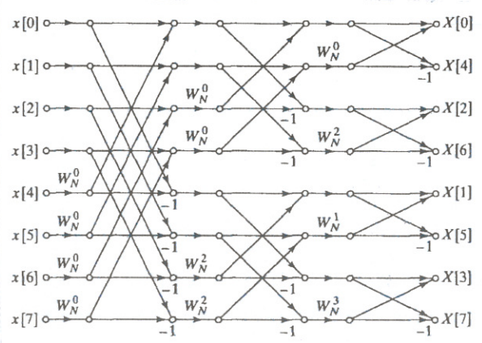

Figure 9.14 Rearrangement of Figure 9.10 with input in normal order and output in bit-reversed order.

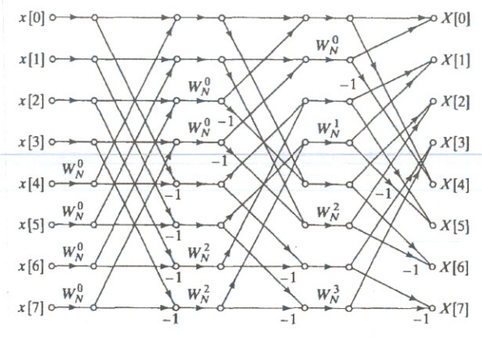

Figure 9.15 Rearrangement of Figure 9.10 with both input and output in normal order.

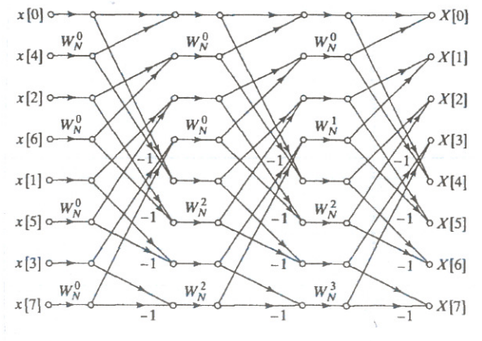

Figure 9.16 Rearrangement of Figure 9.10 having the same geometry for each stage, thereby permitting sequential data accessing and storage.

9.4 DECIMATION-IN-FREQUENCY FFT ALGORITHMS

The decimation-in-time FFT algorithms are all based on structuring the DFT computation by forming smaller and smaller subsequences of the input sequence $x[n]$. Alternatively, we can consider dividing the output sequence $X[k]$ into smaller and smaller subsequences in the same manner. FFT algorithms based on this procedure are commonly called *decimation-in-frequency* algorithms.

To develop these FFT algorithms, let us again restrict the discussion to N a power of 2 and consider computing separately the even-numbered frequency samples and the odd-numbered frequency samples. Since

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \qquad k = 0, 1, \ldots, N-1, \tag{9.23}$$

the even-numbered frequency samples are

$$X[2r] = \sum_{n=0}^{N-1} x[n] W_N^{n(2r)}, \qquad r = 0, 1, \ldots, (N/2)-1, \tag{9.24}$$

which can be expressed as

$$X[2r] = \sum_{n=0}^{(N/2)-1} x[n] W_N^{2nr} + \sum_{n=N/2}^{N-1} x[n] W_N^{2nr}. \tag{9.25}$$

With a substitution of variables in the second summation in Eq. (9.25), we obtain

$$X[2r] = \sum_{n=0}^{(N/2)-1} x[n] W_N^{2nr} + \sum_{n=0}^{(N/2)-1} x[n + (N/2)] W_N^{2r[n+(N/2)]}. \tag{9.26}$$

Finally, because of the periodicity of W_N^{2rn},

$$W_N^{2r[n+(N/2)]} = W_N^{2rn} W_N^{rN} = W_N^{2rn}, \tag{9.27}$$

and since $W_N^2 = W_{N/2}$, Eq. (9.26) can be expressed as

$$X[2r] = \sum_{n=0}^{(N/2)-1} (x[n] + x[n + (N/2)]) W_{N/2}^{rn}, \qquad r = 0, 1, \ldots, (N/2)-1. \tag{9.28}$$

Equation (9.28) is the $(N/2)$-point DFT of the $(N/2)$-point sequence obtained by adding the first half and the last half of the input sequence. Adding the two halves of the input sequence represents time aliasing, consistent with the fact that in computing only the even-numbered frequency samples, we are undersampling the Fourier transform of $x[n]$.

We can now consider obtaining the odd-numbered frequency points, given by

$$X[2r+1] = \sum_{n=0}^{N-1} x[n] W_N^{n(2r+1)}, \qquad r = 0, 1, \ldots, (N/2)-1. \tag{9.29}$$

As before, we can rearrange Eq. (9.29) as

$$X[2r+1] = \sum_{n=0}^{(N/2)-1} x[n] W_N^{n(2r+1)} + \sum_{n=N/2}^{N-1} x[n] W_N^{n(2r+1)}. \tag{9.30}$$

An alternative form for the second summation in Eq. (9.30) is

$$\begin{aligned} \sum_{n=N/2}^{N-1} x[n] W_N^{n(2r+1)} &= \sum_{n=0}^{(N/2)-1} x[n + (N/2)] W_N^{[n+(N/2)](2r+1)} \\ &= W_N^{(N/2)(2r+1)} \sum_{n=0}^{(N/2)-1} x[n + (N/2)] W_N^{n(2r+1)} \\ &= -\sum_{n=0}^{(N/2)-1} x[n + (N/2)] W_N^{n(2r+1)}, \end{aligned} \tag{9.31}$$

where we have used the fact that $W_N^{(N/2)2r} = 1$ and $W_N^{(N/2)} = -1$. Substituting Eq. (9.31) into Eq. (9.30) and combining the two summations, we obtain

$$X[2r+1] = \sum_{n=0}^{(N/2)-1} (x[n] - x[n+(N/2)])W_N^{n(2r+1)}, \tag{9.32}$$

or, since $W_N^2 = W_{N/2}$,

$$X[2r+1] = \sum_{n=0}^{(N/2)-1} (x[n] - x[n+(N/2)])W_N^n W_{N/2}^{nr}, \tag{9.33}$$

$$r = 0, 1, \ldots, (N/2)-1.$$

Equation (9.33) is the $(N/2)$-point DFT of the sequence obtained by subtracting the second half of the input sequence from the first half and multiplying the resulting sequence by W_N^n. Thus, on the basis of Eqs. (9.28) and (9.33), with $g[n] = x[n] + x[n+N/2]$ and $h[n] = x[n] - x[n+N/2]$, the DFT can be computed by first forming the sequences $g[n]$ and $h[n]$, then computing $h[n]W_N^n$, and finally computing the $(N/2)$-point DFTs of these two sequences to obtain the even-numbered output points and the odd-numbered output points, respectively. The procedure suggested by Eqs. (9.28) and (9.33) is illustrated for the case of an 8-point DFT in Figure 9.17.

Proceeding in a manner similar to that followed in deriving the decimation-in-time algorithm, we note that since N is a power of 2, $N/2$ is even; consequently, the $(N/2)$-point DFTs can be computed by computing the even-numbered and odd numbered output points for those DFTs separately. As in the case of the procedure leading to Eqs. (9.28) and (9.33), this is accomplished by combining the first half and the last half of the input points for each of the $(N/2)$-point DFTs and then computing $(N/4)$-point DFTs. The flow graph resulting from taking this step for the 8-point example is shown in Figure 9.18. For the 8-point example, the computation has now been reduced to the computation of 2-point DFTs, which are implemented by adding and subtracting the input points, as discussed previously. Thus, the 2-point DFTs in Figure 9.18 can be replaced by the computation shown in Figure 9.19, so the computation of the 8-point DFT can be accomplished by the algorithm depicted in Figure 9.20.

By counting the arithmetic operations in Figure 9.20 and generalizing to $N = 2^\nu$, we see that the computation of Figure 9.20 requires $(N/2)\log_2 N$ complex multiplications and $N\log_2 N$ complex additions. Thus, the total number of computations is the same for the decimation-in-frequency and the decimation-in-time algorithms.

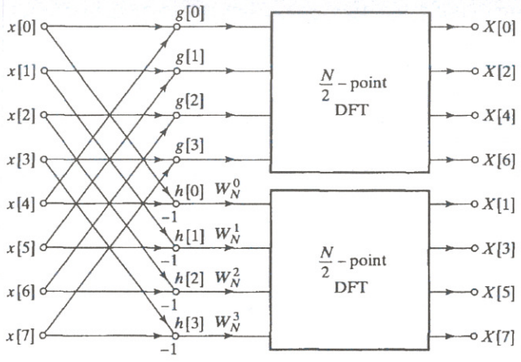

Figure 9.17 Flow graph of decimation-in-frequency decomposition of an N-point DFT computation into two $(N/2)$-point DFT computations ($N = 8$).

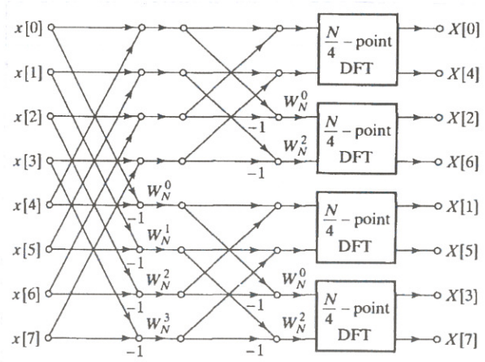

Figure 9.18 Flow graph of decimation-in-frequency decomposition of an 8-point DFT into four 2-point DFT computations.

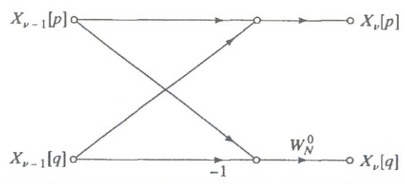

Figure 9.19 Flow graph of a typical 2-point DFT as required in the last stage of decimation-in-frequency decomposition.

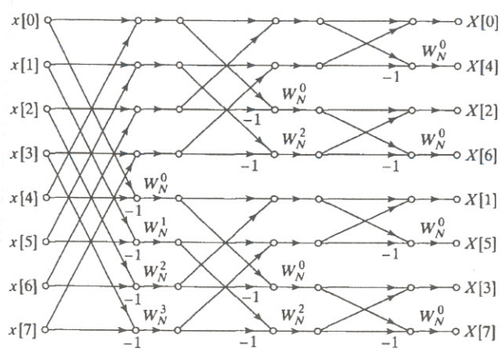

Figure 9.20 Flow graph of complete decimation-in-frequency decomposition of an 8-point DFT computation.

9.4.2 Alternative Forms

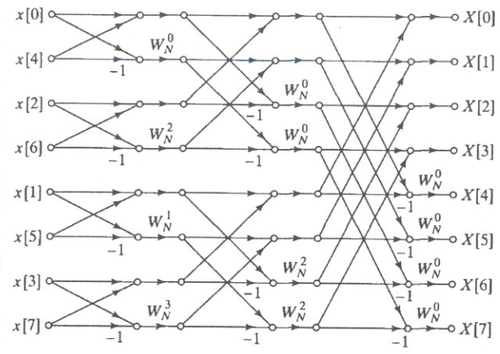

Figure 9.22 Flow graph of a decimation-in-frequency DFT algorithm obtained from Figure 9.20. Input in bit-reversed order and output in normal order. (Transpose of Figure 9.14.)

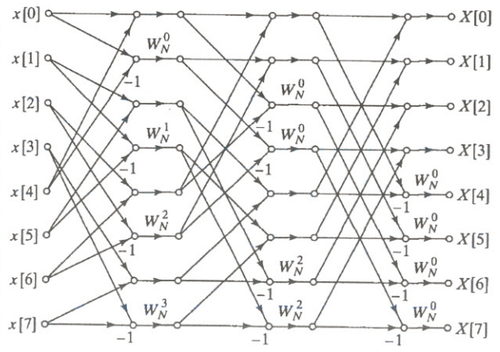

Figure 9.23 Rearrangement of Figure 9.20 with both input and output in normal order. (Transpose of Figure 9.15.)

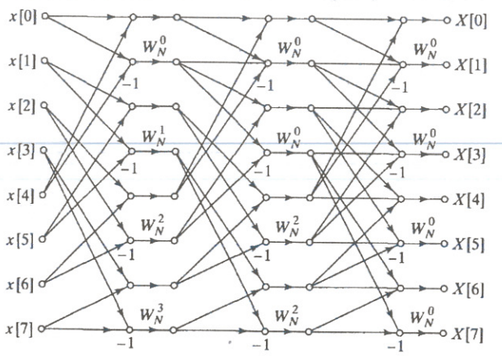

Figure 9.24 Rearrangement of Figure 9.20 having the same geometry for each stage, thereby permitting sequential data accessing and storage. (Transpose of Figure 9.16.)