

The Fast Fourier transforms (FFT)

9.1 Introduction

The time taken to evaluate a DFT on a digital computer depends principally on the number of multiplications involved, since these are the slowest operations. With the DFT, this number is directly related to N^2 , where N is the length of the transform. Therefore for lengthy N the computational speed becomes a major consideration. Derive some fast algorithms for computing the DFT. These algorithms are known, as fast Fourier transforms (FFTs). These algorithms rely on the fact that the standard DFT involves a lot of redundant calculations. , FFT often reducing the computation time by hundreds.

The basic strategy that is used in the FFT algorithm is one of "divide and conquer." which involves decomposing an N -point DFT into successively smaller DFTs.

9.2 Decimation-in-time algorithm

The decimation-in-time FFT algorithm is based on splitting (decimating) $x(n)$ into smaller sequences and finding $X(k)$ from the DFTs of these decimated sequences. This decimation leads to an efficient algorithm when the sequence length is a power of 2. Since the time samples which are divided up, this algorithm is known as the *decimation in-time* (DIT) algorithm.

Let $x(n)$ be a sequence of length $N = 2^v$, and suppose that $x(n)$ is split (decimated) into two subsequences, each of length $N/2$. As illustrated in Fig. 7-1, the first sequence $g(n)$ is formed from the even-index terms,

$$g(n) = x(2r) \quad r = 0, 1, 2, \dots, \frac{N}{2} - 1$$

and the second, $h(n)$, is formed from the odd-index terms,

$$h(n) = x(2r + 1) \quad r = 0, 1, 2, \dots, \frac{N}{2} - 1$$

where $n=2r$ when n is even

$n=2r+1$ when n is odd

For example, consider $N = 8$ (the FFT is simplest by far if N is an integral power of 2)

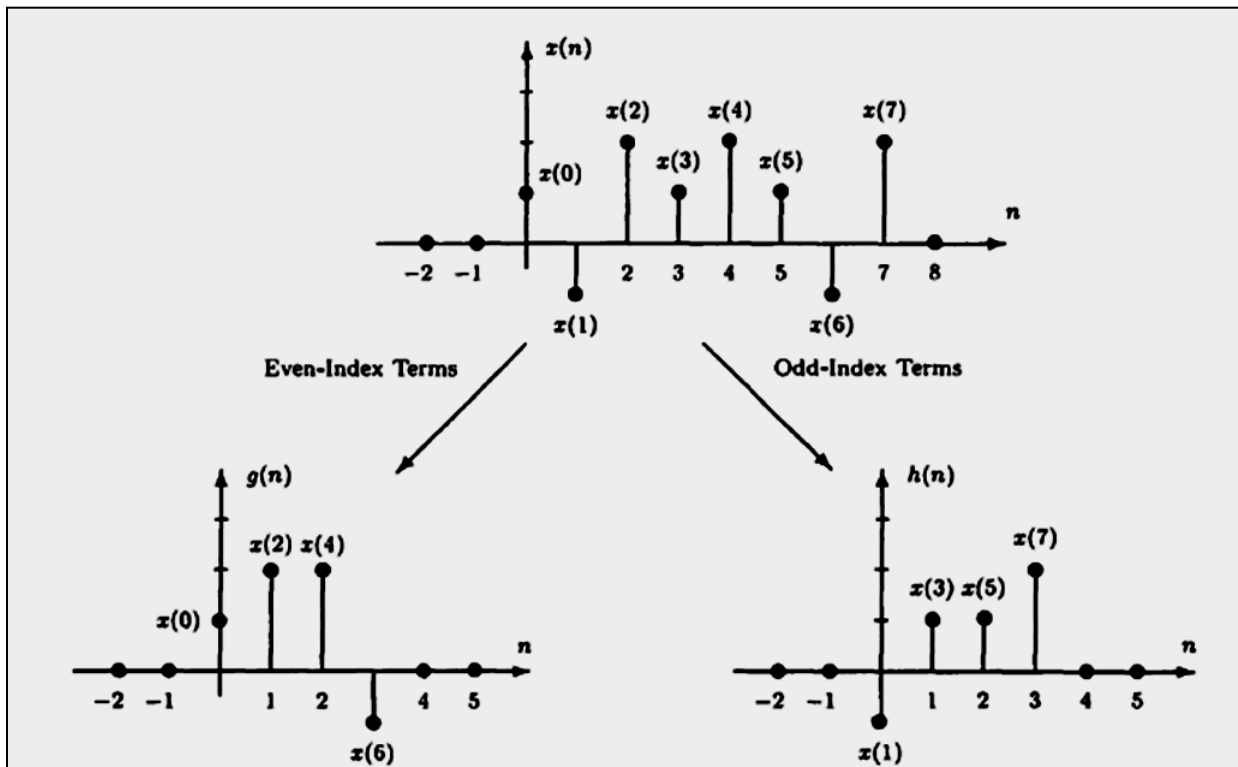


Fig 9-1 Decimation sequences of length $N=8$ by a factor of 2

The N -point DFT of $x(n)$ is

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}$$

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} x(2r) W_N^{2kr} + \sum_{r=0}^{\frac{N}{2}-1} x(2r+1) W_N^{k(2r+1)}$$

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} x(2r) W_N^{\frac{kr}{2}} + \sum_{r=0}^{\frac{N}{2}-1} x(2r+1) W_N^k W_N^{2kr}$$

Where $W_N^2 = W_{N/2}$

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} x(2r) W_N^{\frac{kr}{2}} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x(2r+1) W_N^{\frac{kr}{2}}$$

Note that the first term is the $N/2$ -point DFT of $g(n)$, and the second is the $N/2$ -point DFT of $h(n)$

$$X(k) = G(k) + W_N^k H(k) \quad k = 0, 1, \dots, N-1$$

Where $G(k)$ referred to the transform of the even numbered point in $x(n)$, $H(k)$ referred to the transform of the odd numbered point in $x(n)$

The N - point DFT $X(k)$ can be obtained from two $N/2$ -point transforms. Although the frequency index k ranges over N values, only $N/2$ values of $G(k)$ and $H(k)$ need to be computed since $G(k), H(k)$ are periodic in k with period $N/2$. A block diagram showing the computations that are necessary for the first stage of eight-point decimation-in-time FFT is shown in Fig. 9-2.

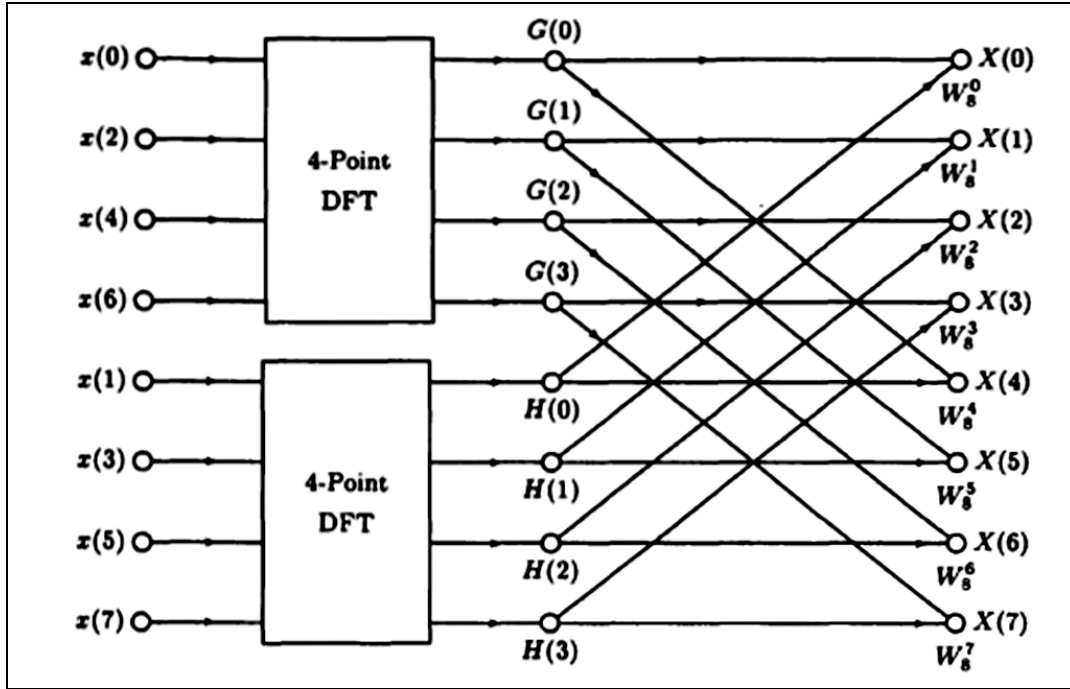


Fig. 9-2. An eight-point decimation-in-time FFT algorithm after the first decimation.

Assuming that N is a power of 2, we can repeat the above process on the two $N/2$ -point transforms, breaking them down to $N/4$ point transforms, etc..., until we come down to 2-point transforms.

Then decimating the sequence to reach to 2-point signal, $G(k)$ may be evaluated as follows:

$$G(k) = \sum_{r=0}^{\frac{N}{4}-1} g(2r) W_{\frac{N}{2}}^{2kr} + \sum_{r=0}^{\frac{N}{4}-1} g(2r+1) W_{\frac{N}{2}}^{k(2r+1)}$$

$$G(k) = \sum_{r=0}^{\frac{N}{2}-1} g(2r) W_{\frac{N}{4}}^{kr} + W_{\frac{N}{2}}^k \sum_{r=0}^{\frac{N}{2}-1} g(2r+1) W_{\frac{N}{4}}^{kr}$$

Where the first term is the $N/4$ -point DFT of the even samples of $g(n)$ and the second is the $N/4$ -point DFT of the odd samples of $g(n)$. A block diagram illustrating this decomposition is shown in Fig. 9-3

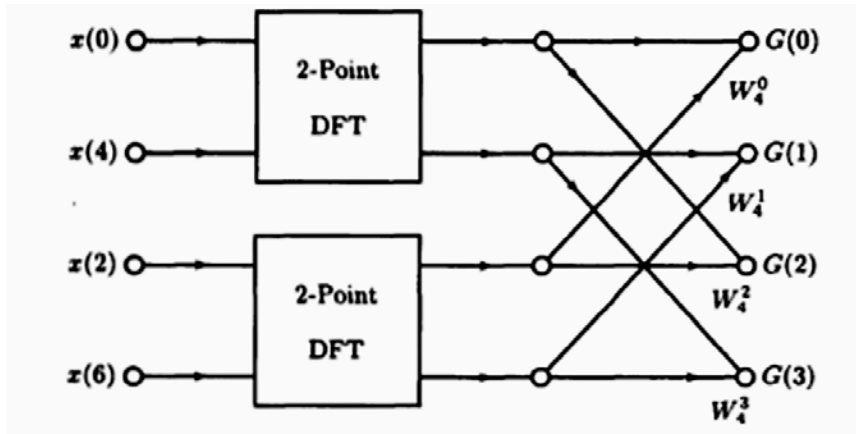


Fig 9-3 Decimation of the four-point DFT into two two-point DFTs in the decimation-in-time FFT.

The basic computation at the heart of the FFT is known as the *butterfly* because of its criss-cross appearance. For the DIT FFT algorithm, the butterfly computation is of the form as below in fig 9-4

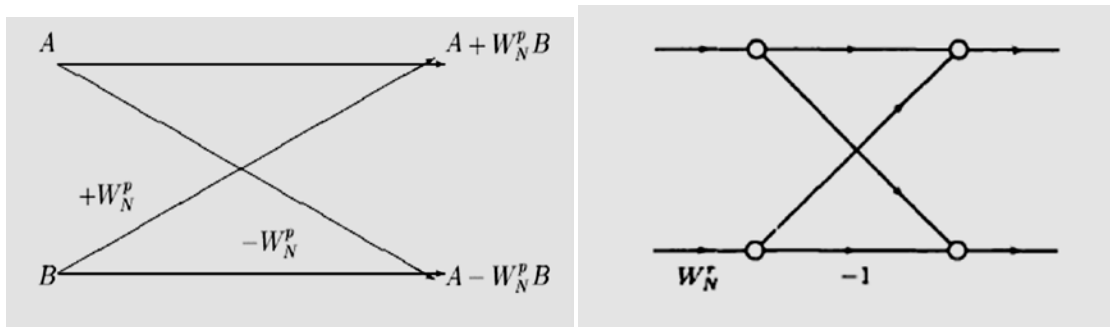


Fig. 9-4 . (a) The butterfly, which is the basic computational element of the FFT algorithm

(b) A simplified butterfly with only one complex multiplication.

The butterfly operation in FFT where A and B are complex numbers. Thus a butterfly computation requires one complex multiplication and 2 complex additions.

A complete eight-point radix-2 decimation-in-time FFT is shown in Fig. 9-5.

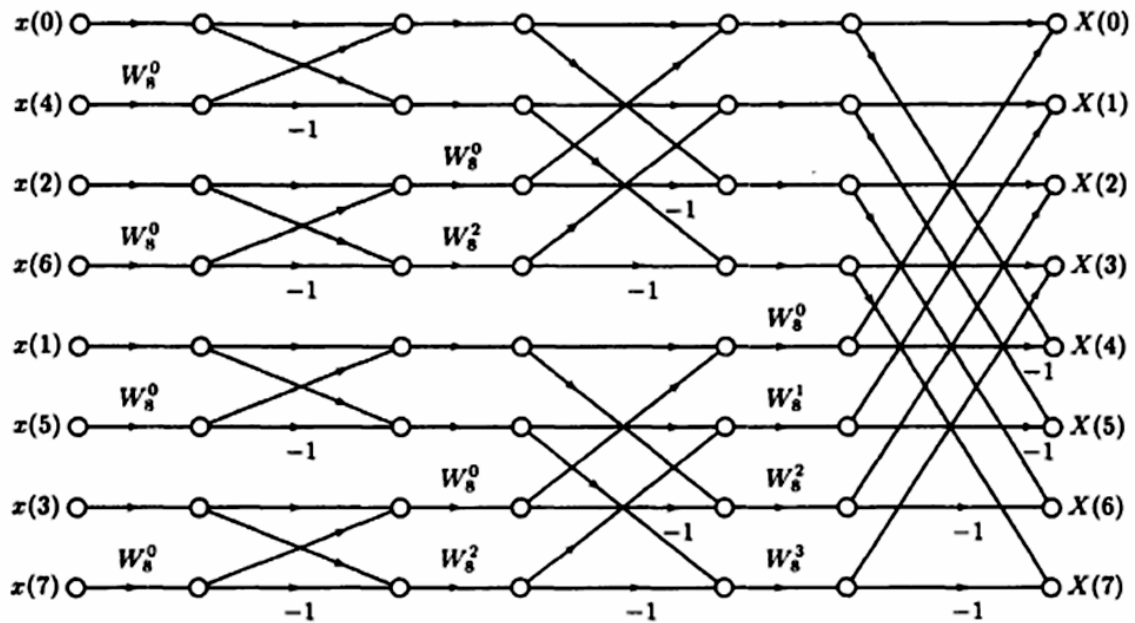
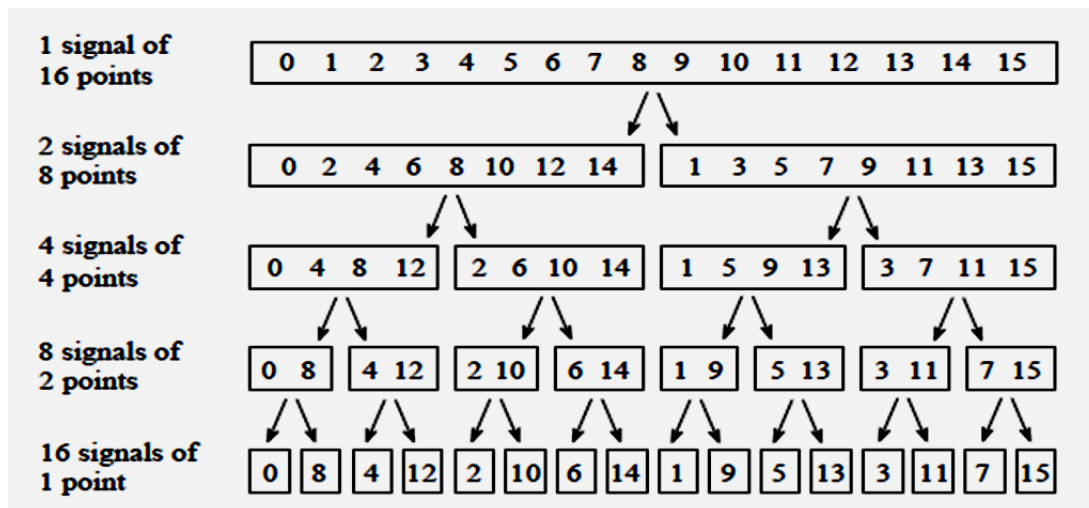


Fig. 9.5 A complete eight-point radix -2, decimation-in-time FFT.

Example of the time domain decomposition used in the FFT. In this example, a 16 point signal is decomposed through four stages.



9.3 Computational speed of FFT

Computing an N -point DFT using radix-2 decimation-in-time FFT is much more efficient than calculating the DFT directly. For example, if $N = 2^v$, there are $\log_2 N = v$ stages of computation. Because each stage requires $N/2$ complex multiplies by the twiddle factors W_N^p and N complex additions. The number of complex multiplications required to evaluate an N -point is $N/2 \log_2 N$ and $N \log_2 N$ complex additions.