EE4C5 Digital Signal Processing

Lecture 14 – The Fast Fourier Transform

This lecture

- Based on Chapter 9 of O&S
- All images from O&S book unless otherwise stated

Refine our definition

- DFT of a finite length sequence with length N

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

- Where $W_N = e^{-j(2\pi/N)}$ (note we have added subscript N here)
- Inverse discrete Fourier Transform:

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

- Consider sequence lengths N which are an integer power of 2
  - $N = 2^v, \;\; v = \log_2 N$
  - Will zero-pad any sequence which is not a power of two

FFT

- The FFT is NOT a separate transform
- An FFT refers to any algorithm which is an efficient (hence “fast”) implementation of the DFT
  - There is more than one FFT
- Major source of efficiency:
  - decomposing the computation of a DFT into successively smaller DFT computations
  - while exploiting both the symmetry and the periodicity of the complex exponential $W_N^{kn} = e^{-j(2\pi/N)kn}$

Decimation in time FFT

Core idea…

- N is divisible by 2
- Consider computing $X[k]$ by separating into two $N/2$ point sequences
  - Even-numbered points $g[n] = x[2n]$
  - Odd-numbered points $h[n] = x[2n+1]$
  - Crucial to notice that original sequence $x[n]$ is just an interleaving of $g[n]$ and $h[n]$

More formally…

- The sequence $x[n]$ is zero for $n < 0$ and for $n > N - 1$. Assume that $N = 2^v$, where M is a positive integer. Let $g[n] = x[2n]$ and $h[n] = x[2n + 1]$.
-
- Prove that the N-point discrete Fourier transform (DFT) of the sequence $x[n]$ can be obtained by appropriately combining the $N/2$-point DFTs of the sequences $g[n]$ and $h[n]$.

- [Done in lectures]
- [$X[k] = G[((k))_{N/2}] + W_N^k H[((k))_{N/2}] \qquad k = 0, 1, \ldots, N - 1$]

Decimation in time, N=8

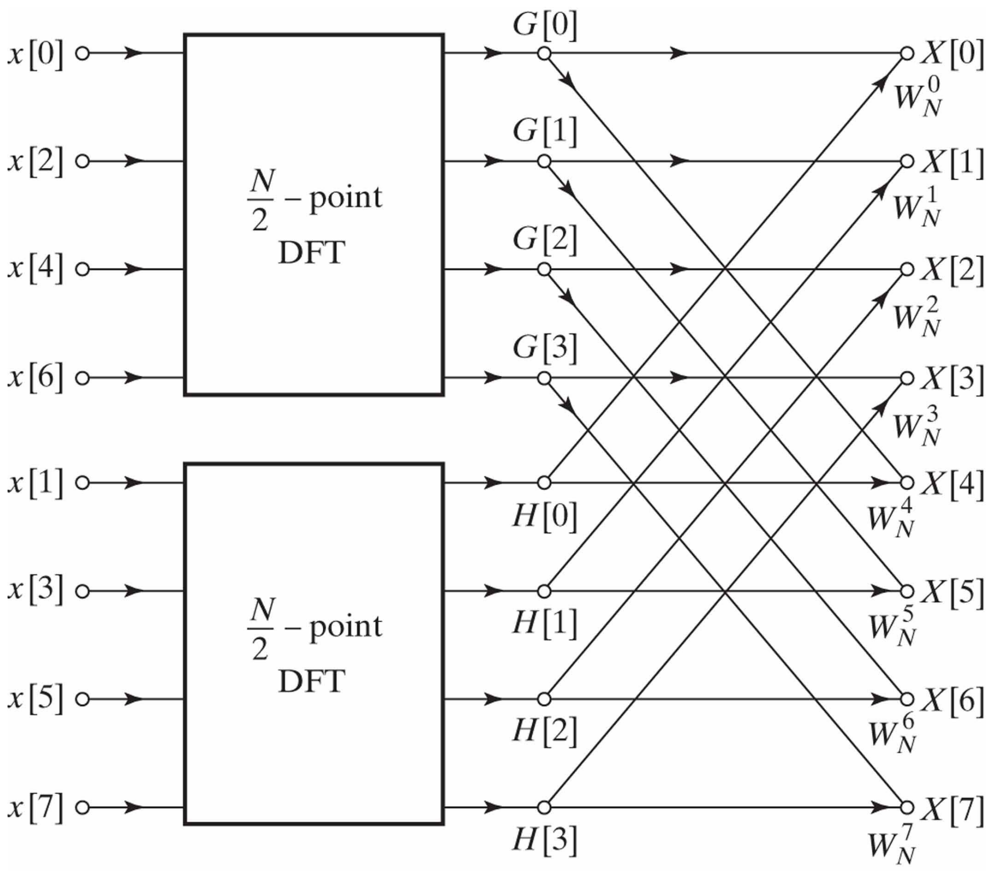


Figure 9.4 Flow graph of the decimation-in-time decomposition of an N-point DFT computation into two ($N/2$)-point DFT computations ($N = 8$).

$$X[k] = G[((k))_{N/2}] + W_N^k H[((k))_{N/2}] \qquad k = 0, 1, \ldots, N-1$$

Can we extend this?

- Yes – we can keep going!
- Can obtain the $N/2$-point DFT $G[k]$ by combining the $N/4$-point DFTs of $g[2l]$ and $g[2l+1]$

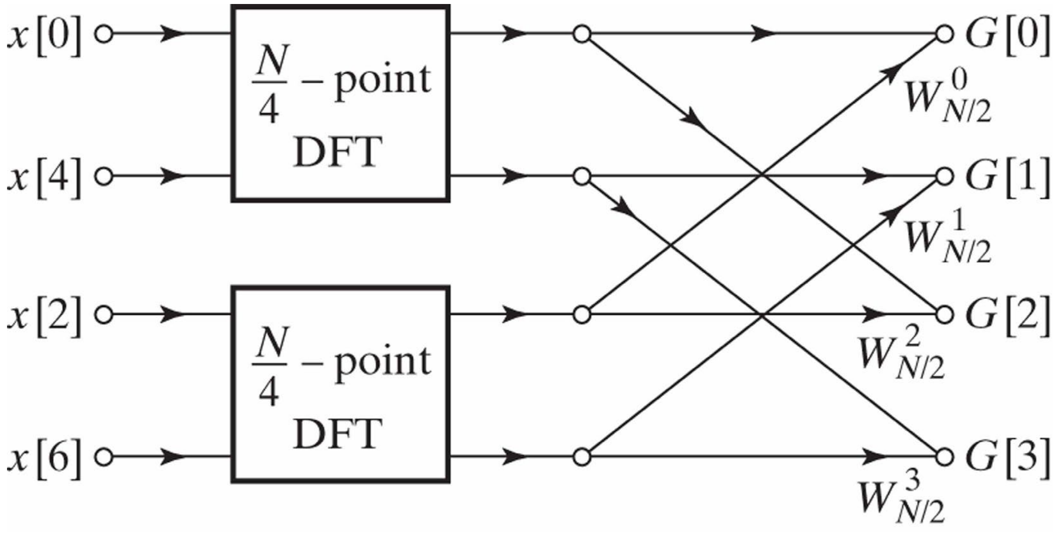


Figure 9.5 Flow graph of the decimation-in-time decomposition of an ($N/2$)-point DFT computation into two ($N/4$)-point DFT computations (N = 8).

Combining these

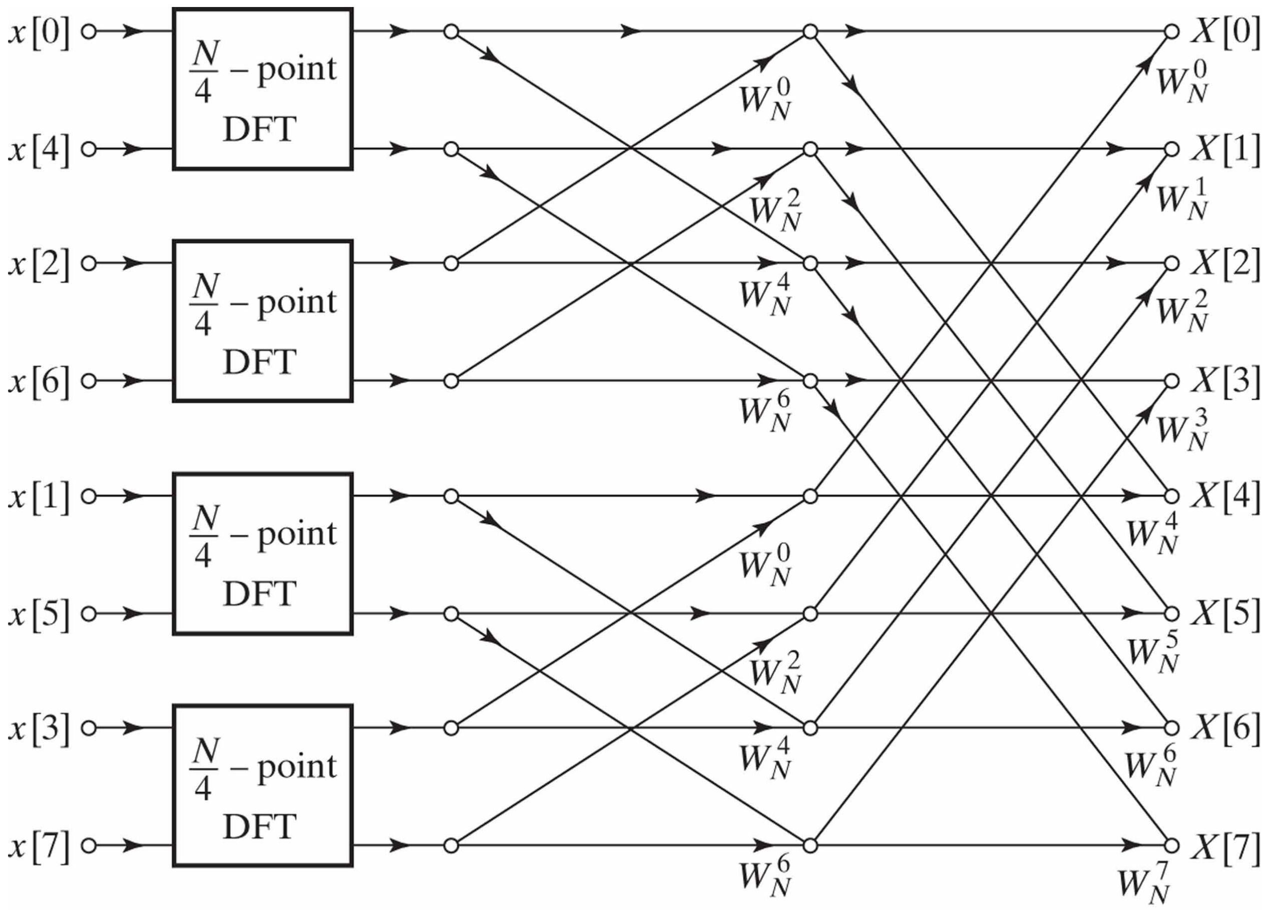


Figure 9.6 Result of substituting the structure of Figure 9.5 into Figure 9.4.

Keep extending…

- If we continue until we are left with just 2-point DFT (i.e. $N = 2$)
- Will have $v = \log_2 N$ stages

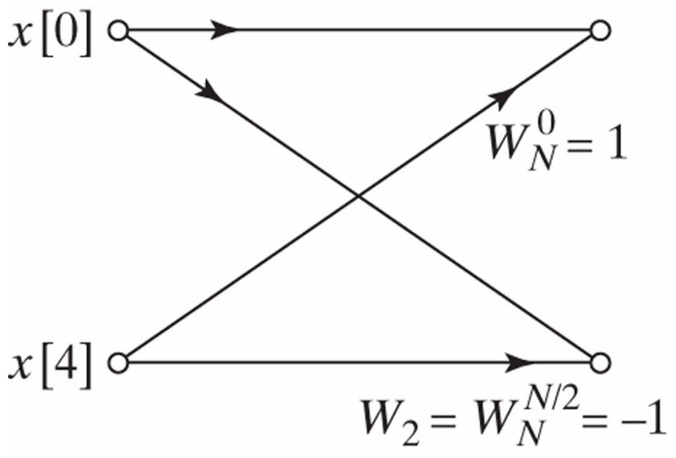


Figure 9.7 Flow graph of a 2-point DFT.

Complete flow-graph for 8-point DFT

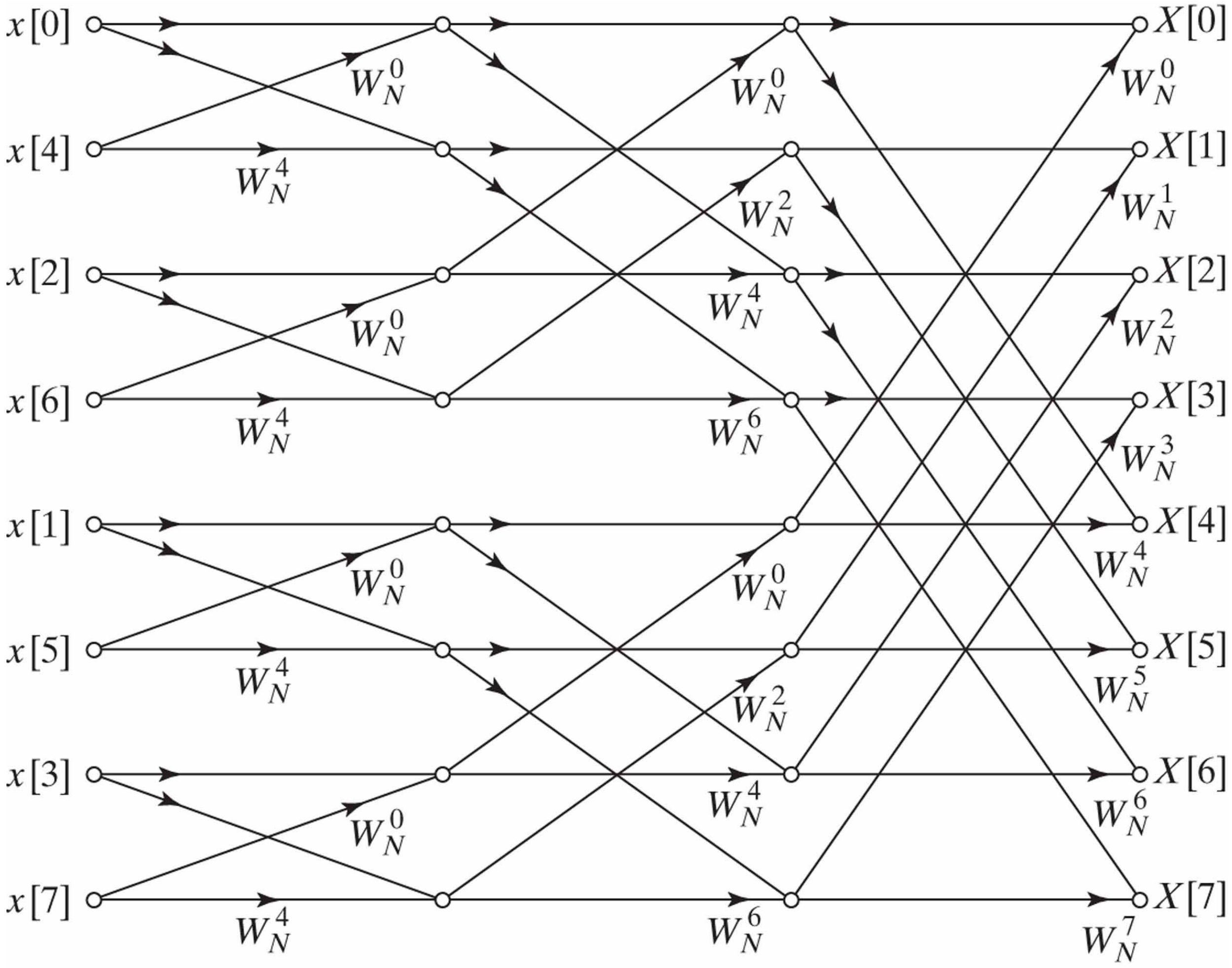


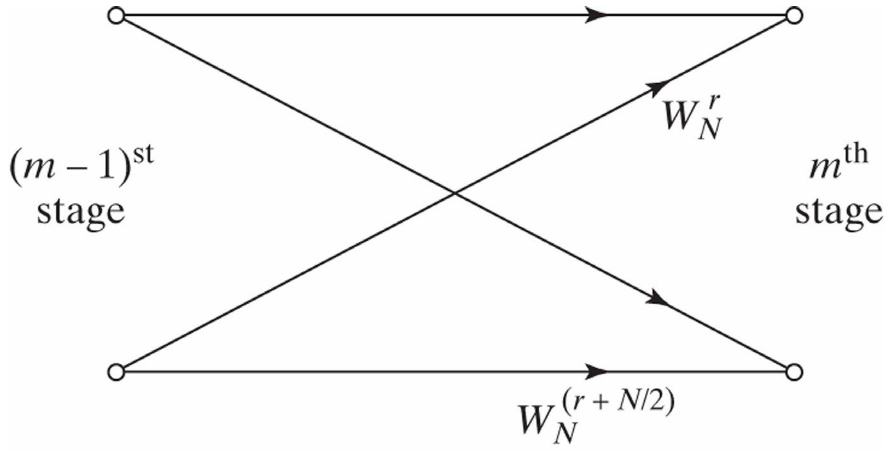
Figure 9.9 Flow graph of complete decimation-in-time decomposition of an 8-point DFT computation.

Decimation-in-Time Radix-2 FFT Algorithm

Basic Butterfly Computation

- Basic butterfly has 2 complex multiplications
- Obtain a pair of values in one stage from a pair of values in the preceding stage
  - the coefficients are always powers of W_N and the exponents are separated by $N/2$.

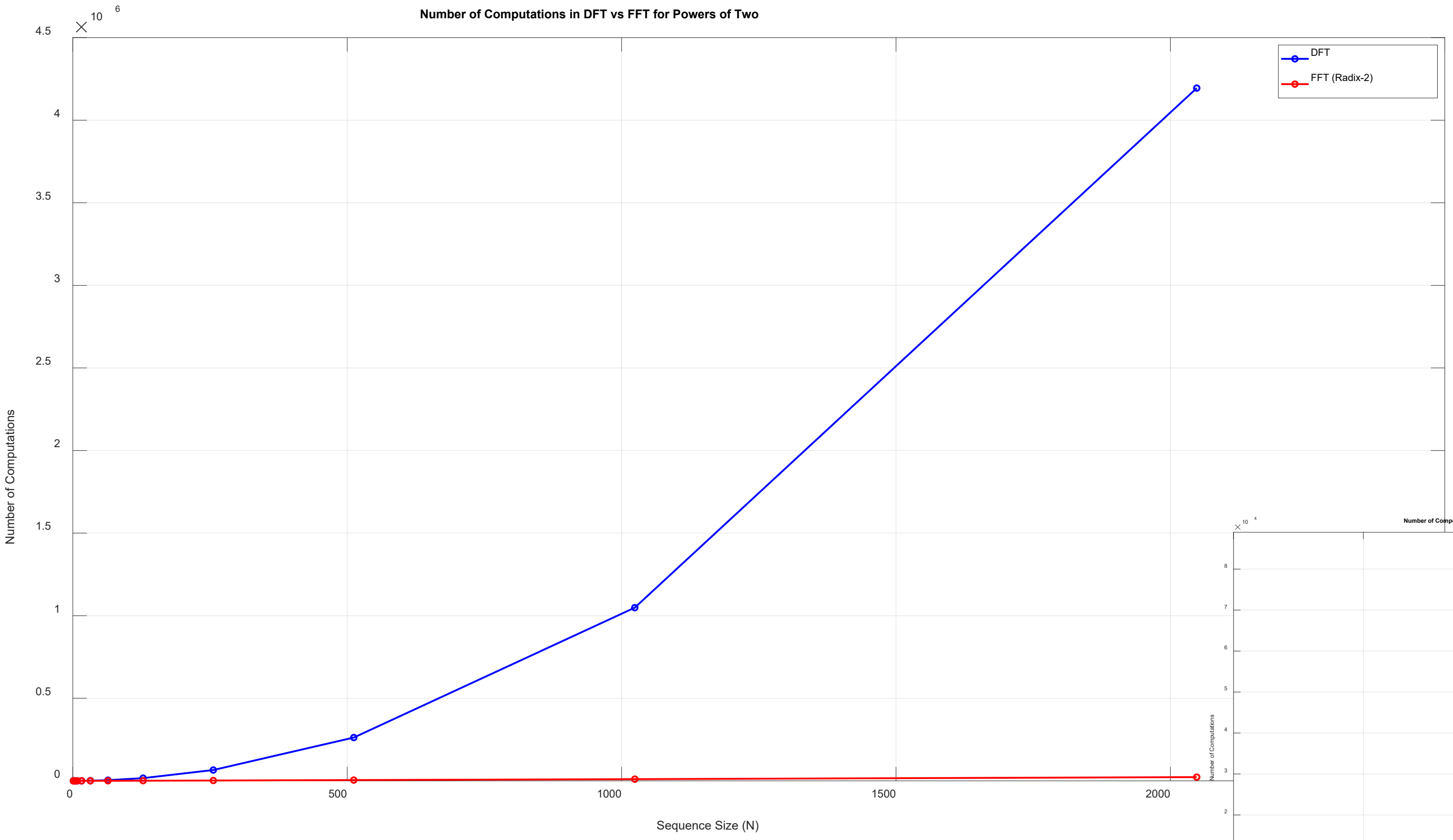
Figure 9.8 Flow graph of basic butterfly computation in Figure 9.9.

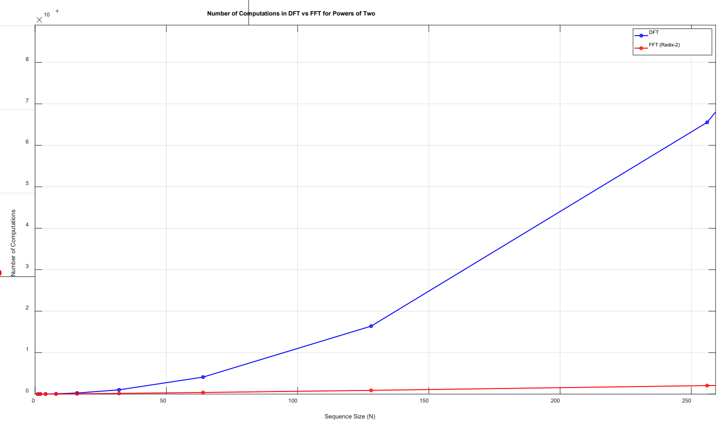


Order of computations?

- Each stage has
  - N complex multiplications
  - N complex additions
- With $\log_2 N$ stages
  - Have $N\log_2 N$ complect multiplications and additions

FFT versus DFT – computations





Further refinements

- Computation in the flow graph (labelled Figure 9.9 earlier) can be reduced further by exploiting the symmetry and periodicity of the coefficients W_N^r
- Observe that: $$W_N^{N/2} = e^{-j(2\pi/N)N/2} = e^{-j\pi} = -1$$
- Hence factor $W_N^{r+N/2}$ can be written as:
$$W_N^{r+N/2} = W_N^{N/2} W_N^r = -W_N^r$$
- Need one complex addition and one complex subtraction, but only one complex multiplication instead of two
  - Reduction of multiplies by factor of two

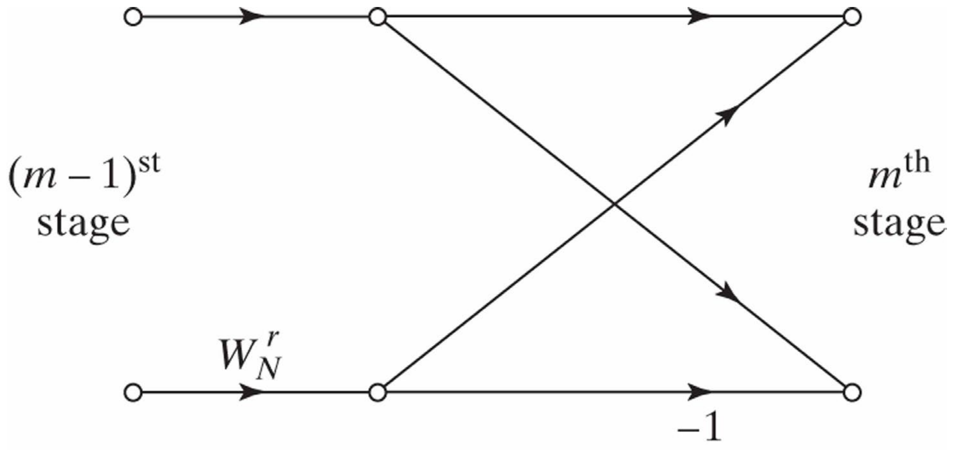


Figure 9.10 Flow graph of simplified butterfly computation requiring only one complex multiplication.

Include these refinements

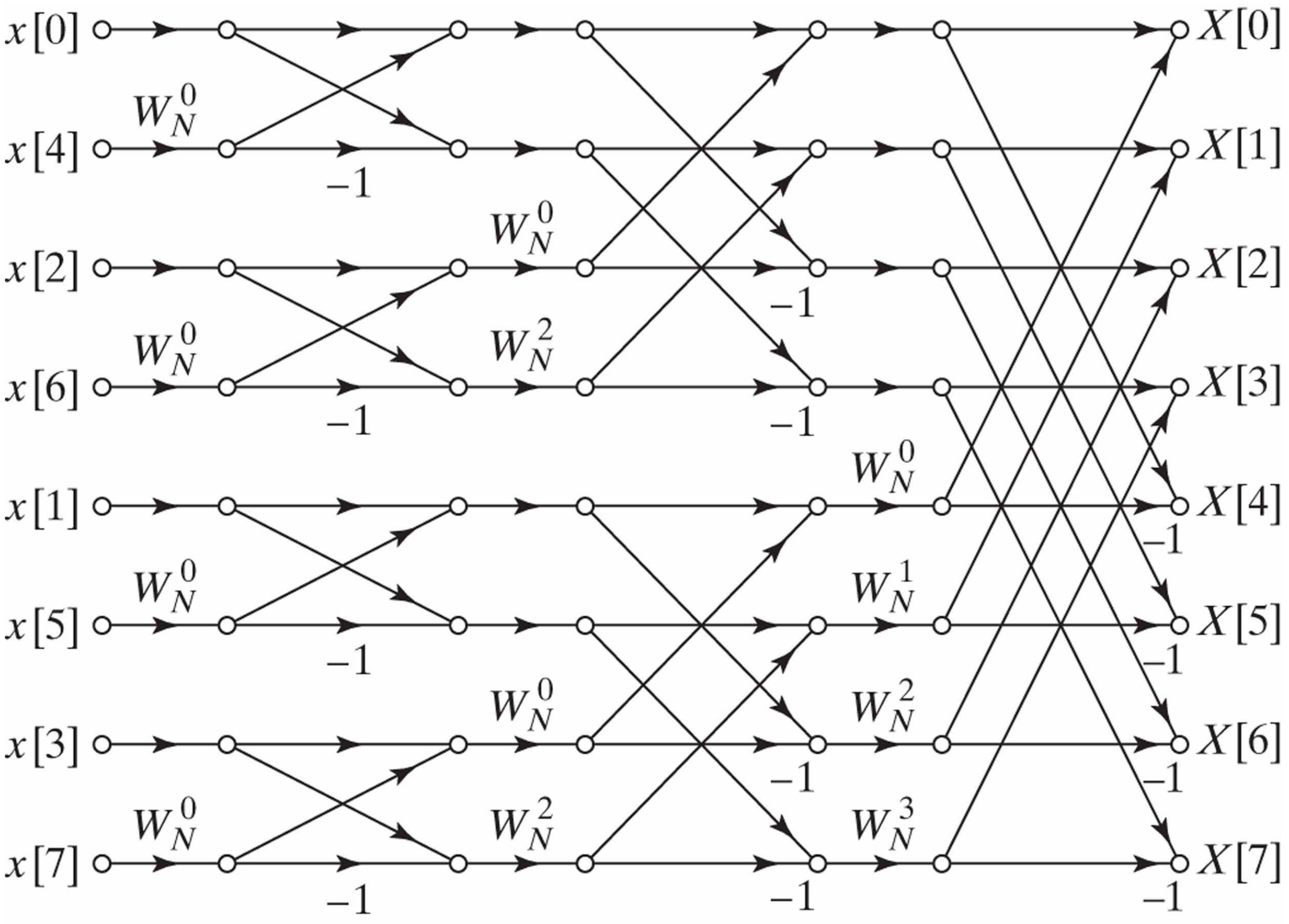
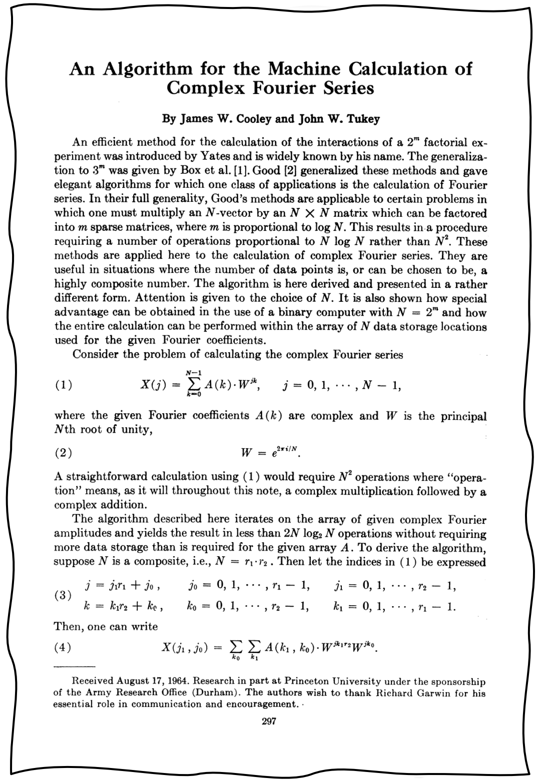


Figure 9.11 Flow graph of 8-point DFT using the butterfly computation of Figure 9.10.

Cooley and Tukey

- Many forms of the FFT
- Orignal paper uses graph on right here



An Algorithm for the Machine Calculation of Complex Fourier Series

By James W. Cooley and John W. Tukey

An efficient method for the calculation of the interactions of a 2^m factorial experiment was introduced by Yates and is widely known by his name. The generalization to 3^m was given by Box et al. [1]. Good [2] generalized these methods and gave elegant algorithms for which one class of applications is the calculation of Fourier series. In their full generality, Good's methods are applicable to certain problems in which one must multiply an N-vector by an $N \times N$ matrix which can be factored into m sparse matrices, where m is proportional to $\log N$. This results in a procedure requiring a number of operations proportional to $N \log N$ rather than N^2. These methods are applied here to the calculation of complex Fourier series. They are useful in situations where the number of data points is, or can be chosen to be, a highly composite number. The algorithm is here derived and presented in a rather different form. Attention is given to the choice of N. It is also shown how special advantage can be obtained in the use of a binary computer with $N = 2^m$ and how the entire calculation can be performed within the array of N data storage locations used for the given Fourier coefficients.

Consider the problem of calculating the complex Fourier series

$$X(j) = \sum_{k=0}^{N-1} A(k) \cdot W^{jk}, \qquad j = 0, 1, \cdots, N-1, \tag{1}$$

where the given Fourier coefficients $A(k)$ are complex and W is the principal Nth root of unity,

$$W = e^{2\pi i/N}. \tag{2}$$

A straightforward calculation using (1) would require N^2 operations where "operation" means, as it will throughout this note, a complex multiplication followed by a complex addition.

The algorithm described here iterates on the array of given complex Fourier amplitudes and yields the result in less than $2N \log_2 N$ operations without requiring more data storage than is required for the given array A. To derive the algorithm, suppose N is a composite, i.e., $N = r_1 \cdot r_2$. Then let the indices in (1) be expressed

$$\begin{aligned} j &= j_1 r_1 + j_0, & j_0 &= 0, 1, \cdots, r_1 - 1, & j_1 &= 0, 1, \cdots, r_2 - 1, \\ k &= k_1 r_2 + k_0, & k_0 &= 0, 1, \cdots, r_2 - 1, & k_1 &= 0, 1, \cdots, r_1 - 1. \end{aligned} \tag{3}$$

Then, one can write

$$X(j_1, j_0) = \sum_{k_0} \sum_{k_1} A(k_1, k_0) \cdot W^{jk_1 r_2} W^{jk_0}. \tag{4}$$

Received August 17, 1964. Research in part at Princeton University under the sponsorship of the Army Research Office (Durham). The authors wish to thank Richard Garwin for his essential role in communication and encouragement.

297

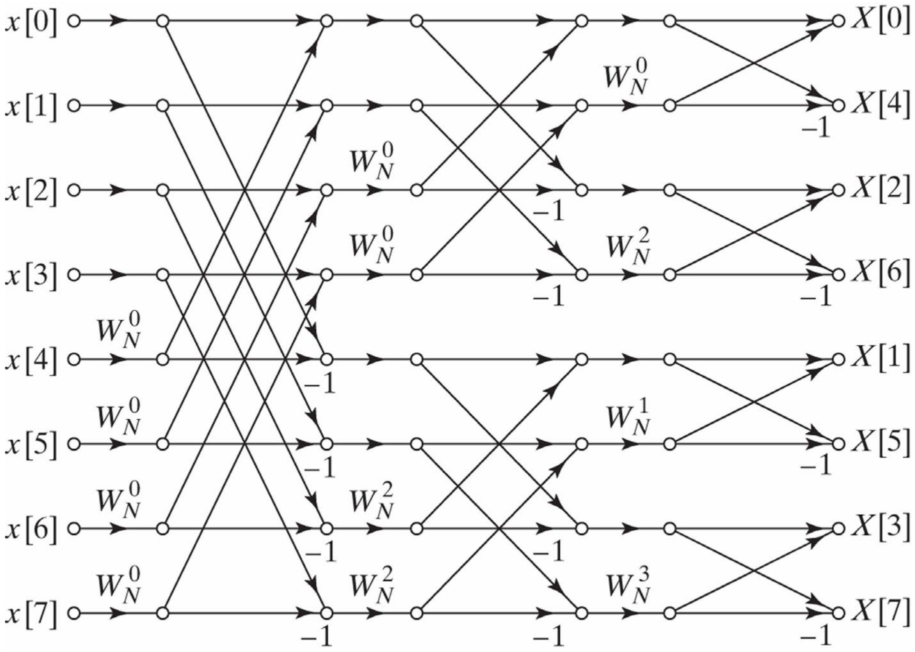


Figure 9.15 Rearrangement of Figure 9.11 with input in normal order and output in bit-reversed order.

Applications

- Spectral Analysis:
  - Use FFT to convert a time-domain signal into its frequency components.
  - Identify peak frequencies, amplitudes, and phase information for detailed spectral analysis.
- Filtering:
  - Implement Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters using FFT.
  - Optimize filter design by analysing filter characteristics in the frequency domain.
- Audio Processing:
  - Employ FFT for real-time audio spectrum analysis in applications like equalizers.
  - Implement audio compression algorithms, such as MP3, which rely on frequency domain representation.
- Image Processing:
  - Apply 2D FFT for image filtering in spatial frequency domain.
  - Achieve image compression through transformations like the Discrete Cosine Transform (DCT).
- Communication Systems:
  - Use FFT for modulation schemes like Orthogonal Frequency Division Multiplexing (OFDM).
  - Perform signal demodulation and channel equalization using FFT algorithms.

More applications

- Radar Systems:
    - Utilize FFT for pulse compression, enabling radar systems to distinguish between targets at different ranges.
    - Implement Doppler processing to analyze the frequency shift caused by moving targets.
- Biomedical Signal Processing:
    - Analyse Electroencephalogram (EEG) signals using FFT to identify frequency patterns associated with brain activity.
    - Extract heart rate information from Electrocardiogram (ECG) signals through frequency domain analysis.
- Speech Processing:
    - Apply FFT for speech feature extraction, including Mel Frequency Cepstral Coefficients (MFCCs).
    - Implement pitch detection algorithms based on the frequency content of speech signals.
- Vibration Analysis:
    - Use FFT to convert time-domain vibration signals into frequency-domain representations.
    - Identify resonance frequencies and analyse vibrational modes in mechanical systems.
- Power Analysis:
    - Monitor power quality by analysing harmonic components in electrical signals using FFT.
    - Detect and diagnose power system faults based on frequency domain analysis of voltage and current waveforms.

Required Reading & other material

- Oppenheim & Schafer, Chapter 9
- Approachable paper: Kumar, G. Ganesh, Subhendu K. Sahoo, and Pramod Kumar Meher. "50 years of FFT algorithms and applications." *Circuits, Systems, and Signal Processing* 38 (2019): 5665-5698.
- "The FFT - an algorithm the whole family can use" https://www.cs.dartmouth.edu/~rockmore/cse-fft.pdf