COMPUTATION OF DISCRETE FOURIER TRANSFORM - PART 2

1. Lecture 19 - 49 minutes

a.

Decimation in Frequency

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

$$=\sum_{n=0}^{\frac{N}{2}-1}x(n)W_N^{nk}+\sum_{n=\frac{N}{2}}^{N-1}x(n)W_N^{nk}$$

$$W_N^{\frac{N}{2}k}\sum_{n=0}^{\frac{N}{2}-1}x(n+\tfrac{N}{2})W_N^{nk}$$

$$X(k)=\sum_{n=0}^{\frac{N}{2}-1}\left[x(n)+(-1)^k x(n+\tfrac{N}{2})\right]W_N^{nk}$$

k even - $X(2r)$

$$\sum_{n=0}^{\frac{N}{2}-1}\left[x(n)+x(n+\tfrac{N}{2})\right]W_N^{2rn}$$

$r=0,1,2\ldots(\frac{N}{2}-1)$

k odd - $X(2r+1)$

$r=0,1,\ldots(\frac{N}{2}-1)$

$$\sum_{n=0}^{\frac{N}{2}-1}\left[x(n)-x(n+\tfrac{N}{2})\right]W_N^{n}W_N^{2rn}$$

$W_N^{2rn}=W_{\frac{N}{2}}^{rn}$

b.

k even

$$X(2r)=\sum_{n=0}^{\frac{N}{2}-1}g(n)W_{N/2}^{rn}$$

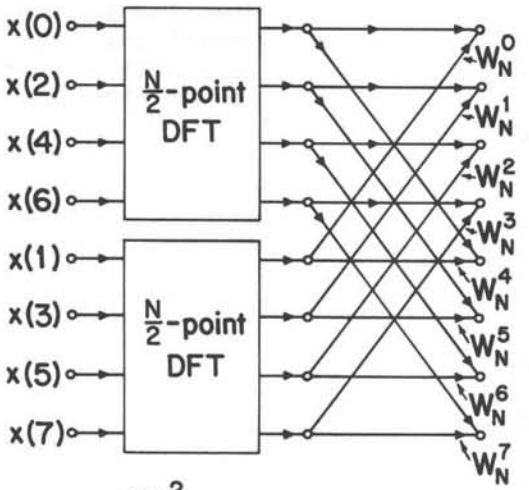
$$g(n)=x(n)+x(n+\tfrac{N}{2})$$

k odd

$$X(2r+1)=\sum_{n=0}^{\frac{N}{2}-1}\left[h(n)W_N^n\right]W_{N/2}^{rn}$$

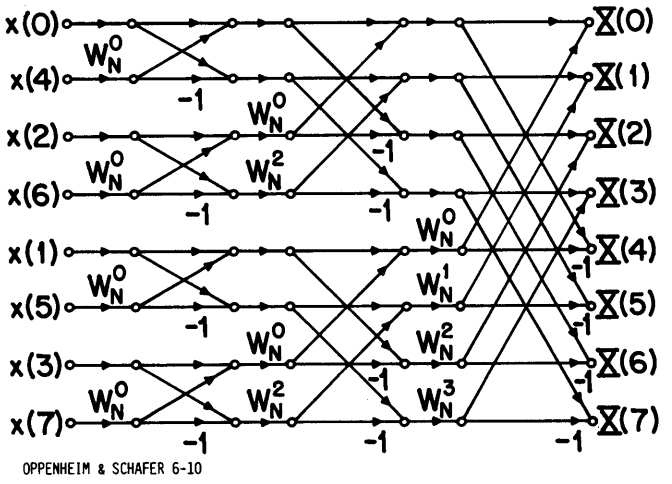
$$h(n)=x(n)-x(n+N/2)$$

c.



$2\left(\frac{N}{2}\right)^2+N$ MADS

Decomposition of an N-point DFT into 2 N/2-point DFT's.



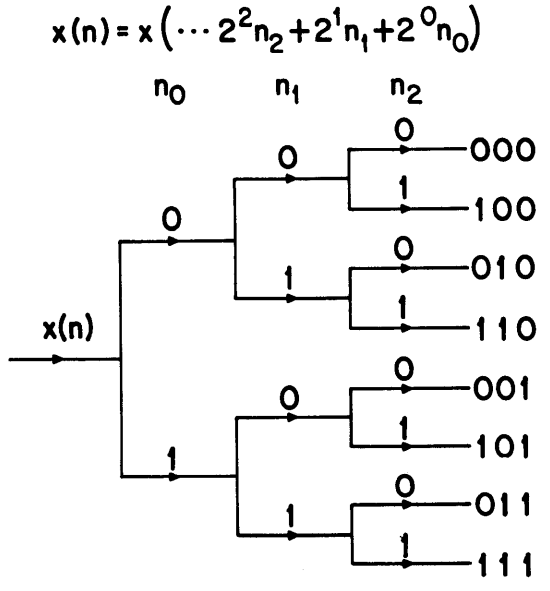
FFT flow-graph for decimation-in-time algorithm.

d.

	Storage Register		Data Index
0	000	X(0)	000
1	001	X(4)	100
2	010	X(2)	010
3	011	X(6)	110
4	100	X(1)	001
5	101	X(5)	101
6	110	X(3)	011
7	111	X(7)	111

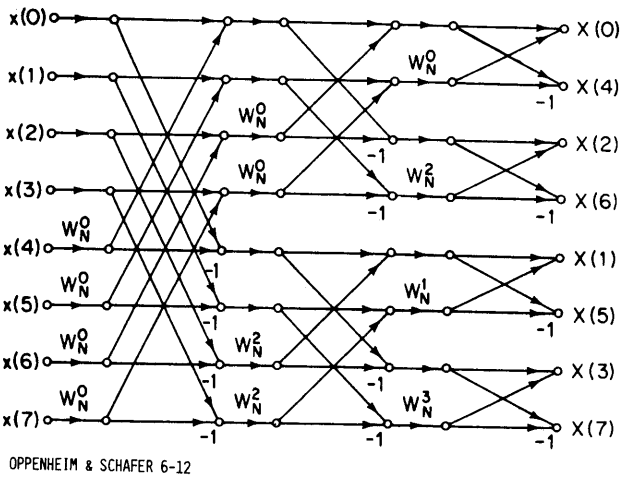
Relation between data index and data storage register.

e.



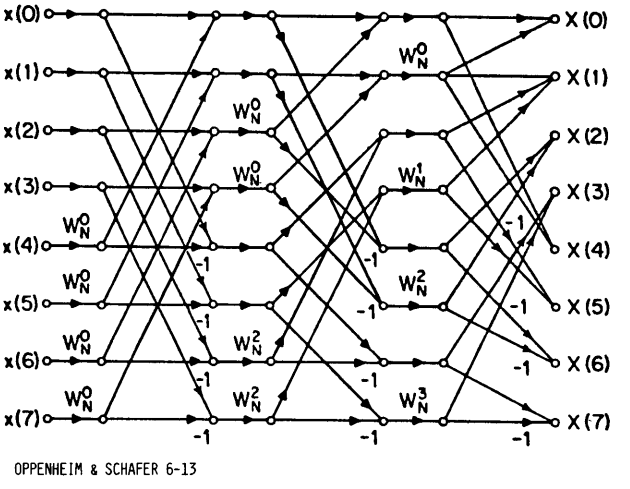
Sorting of data implied by the development of the decimation-in-time algorithm.

f.



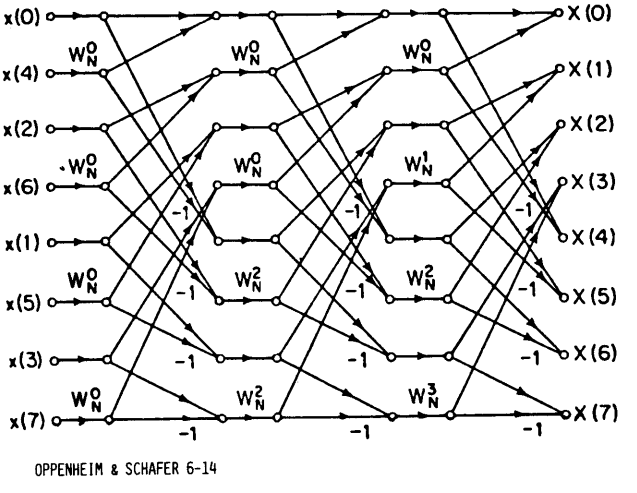
g.

Rearrangement of flow-graph d. with data in normal order and output in bit-reversed order.

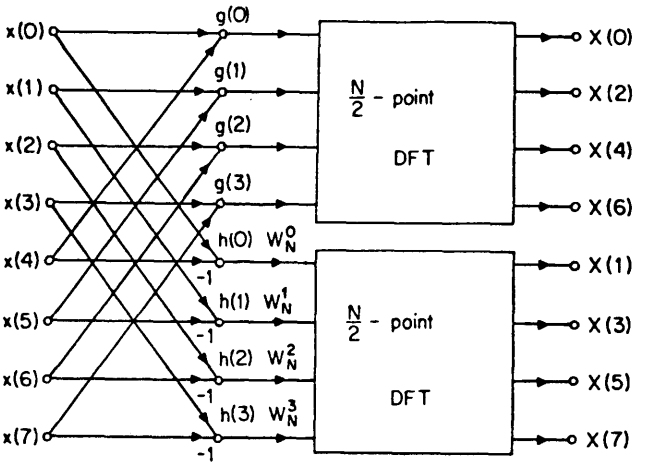


h.

Rearrangement of flow-graph d having the same geometry for each stage.

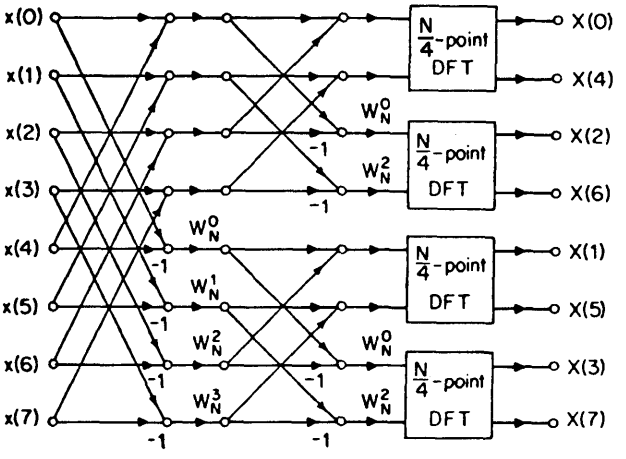


i.



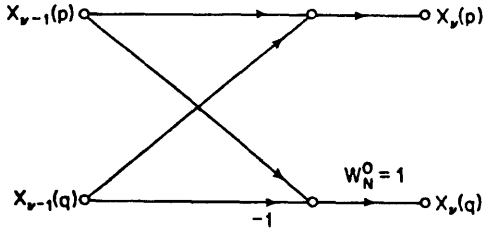
Computation of even and odd-numbered DFT values.

j.



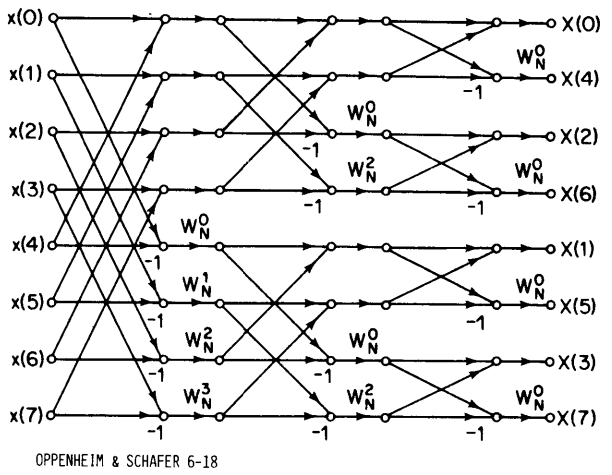
Decomposition of the N/2-point DFT's of flow-graph j.

k.



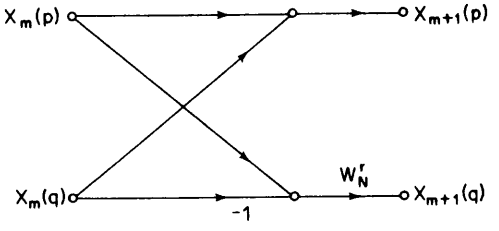
Flow-graph for two-point DFT.

l.



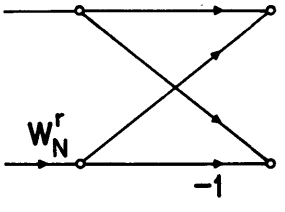
Flow-graph of complete decimation-in-frequency decomposition of an eight-point DFT computation.

m.



Flow-graph of a typical butterfly computation required in decimation-in-frequency FFT algorithm.

n.



Flow-graph of a typical butterfly computation required in decimation-in-time FFT algorithm.

o.

2. Comments

In this lecture we continue the discussion which was begun last time, in which we had developed a flow chart for efficient computation of the DFT. Here we discuss the implication of that flow chart for "in-place" computation if the input data is stored in a "bit reversed" order. This bit-reversed order arises as a natural consequence of the way in which that flow-graph was derived, i.e. by dividing the input into its even and odd numbered points, then dividing each of those in a similar manner, etc.

By rearranging the flow-graph it is possible to generate a number of other algorithms. For example, the input ordering can be rearranged so that bit reversal is not required. However, the advantage of in-place computation is lost. Another rearrangement, commonly referred to as the Singleton algorithm results in a flow-graph structure which is identical from stage to stage. This form is useful when the data is stored in sequential memory such as shift registers or disk rather than random access memory.

The above algorithms are collectively referred to as "decimation-in-time" algorithms because they were based on successive subdivisions of the input sequence. A companion set of algorithms referred to as "decimation-in-frequency" are based on successive subdivisions of the output. We conclude this lecture by introducing this class of algorithms, obtaining in particular, the basic flow-graph representing this algorithm. In the next lecture we consider a number of rearrangements of this flow-graph.

3. Reading

Text: Section 9.3.1 (page 592) and 9.3.2. Also section 9.4 up to, but not including section 9.4.1.

4. Problems

Problem 19.1

Consider a 16-point sequence x(0), x(1),..., x(15). List the sequence in bit-reversed order.

Problem 19.2

(a) Draw the flow-graph for a four-point decimation-in-time FFT algorithm utilizing the butterflies of Figure 9.9 of the text and with the input in bit-reversed order, the output in normal order, and

representing in-place computation.

(b) Rearrange the flow graph of part (a) so that it still corresponds to in-place computation but with the input in normal-order and the output in bit-reversed order.

Problem 19.3

Consider the FFT algorithm for N a power of 2, implemented in the form characterized by Figure 9.20 (page 602) in the text. We are assuming that N is an arbitrary power of 2, not that N=8. In indexing through the data in an array, we shall assume that points in an array are stored in consecutive complex (double) registers numbered 0 through N - 1. The arrays are numbered 0 through $\log_2 N$. The array holding the initial data is the zeroth array. The output of the first stage of butterflies is the first array, etc. All of the following questions relate to the computation of the m^{th} array where $1 \leq m \leq \log_2 N$. The answer should be in terms of m. All of the questions can be answered by generalizing the results for N = 8.

(a) How many butterflies are to be computed?

(b) What are the powers of W_N involved in computing the m^{th} array from the (m - 1)st array?

(c) What is the separation between the addresses of the two complex input points to a butterfly?

(d) What is the separation between the addresses of the first points of butterflies utilizing the same coefficients? Note that the butterfly computation for this algorithm is of the form of Fig. 9.21 in the text, i.e. the coefficient multiplication is applied at the output of the butterfly.

Problem 19.4*

When implementing a decimation-in-time FFT algorithm, the basic butterfly computation is as shown in the flow graph of Figure P19.4-1

$$X_{m+1}(p) = X_m(p) + W_N^r X_m(q)$$

$$X_{m+1}(q) = X_m(p) - W_N^r X_m(q)$$

In using fixed-point arithmetic in implementing the computations it is commonly assumed that all numbers are scaled to be less than unity. Thus we must be concerned with overflow in the butterfly computations.

(a) Show that if we require

$|X_m(p)| < \frac{1}{2}$ and $|X_m(q)| < \frac{1}{2}$

then overflow cannot occur in the butterfly computation; i.e.,

$\text{Re}\,[X_{m+1}(p)] < 1$, $\text{Im}\,[X_{m+1}(p)] < 1$, $\text{Re}\,[X_{m+1}(q)] < 1$,

and $\text{Im}\,[X_{m+1}(q)] < 1$

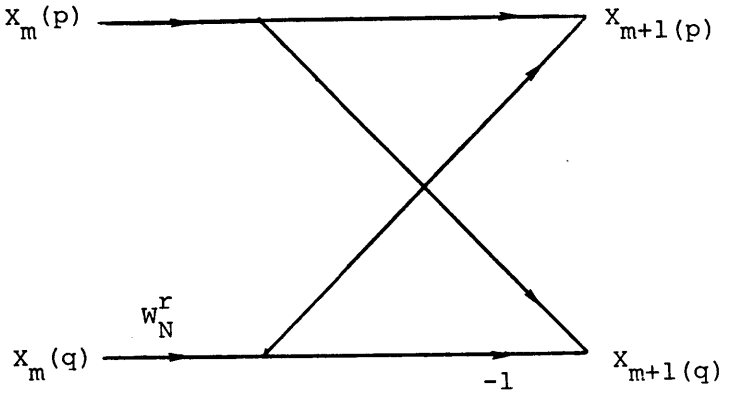


Figure P19.4-1

(b) In practice, it is easier and more convenient to require

$$|\text{Re}[X_m(p)]| < \frac{1}{2}, \quad |\text{Im}[X_m(p)]| < \frac{1}{2}$$

$$|\text{Re}[X_m(q)]| < \frac{1}{2}, \quad |\text{Im}[X_m(q)]| < \frac{1}{2}$$

Are these conditions sufficient to ensure that overflow cannot occur in the butterfly computation? Justify your answer.

Problem 19.5*

The FORTRAN program shown below implements the decimation-in-time algorithm of Figure 9.10 of the text. In the subroutine FFT(X, M), X is a complex array of dimension N that contains initially the input sequence x(n) and finally contains the transform X(k). The quantity M is an integer, $M = \log_2 N$.

(a) From a cursory inspection of the program indicate which lines of code are concerned with (1) bit reversal, (2) recursive computation of the complex exponential multipliers, and (3) the basic butterfly computation.

(b) Three errors have been inserted into the program as it is given here. Find these errors and make appropriate corrections to the FORTRAN code.

```
      SUBROUTINE FFT(X,M)                                              0001
      COMPLEX X(1024), U,W,T                                           0002
      N = 2**M                                                         0003
      NV2 = N/2                                                        0004
      NM1 = N - 1                                                      0005
      J = 1                                                            0006
      DO 7 I = 1,NM1                                                   0007
      T = X(J)                                                         0008
      X(J) = X(I)                                                      0009
      X(I) = T                                                         0010
5     K = NV2                                                          0011
6     IF(K.GE.J) GO TO 7                                               0012
      J = J - K                                                        0013
      K = K/2                                                          0014
      GO TO 6                                                          0015
7     J = J + K                                                        0016
      PI = 3.14159265358979                                            0017
      DO 20 L = 1,M                                                    0018
      LE = 2**L                                                        0019
      LE1 = LE/2                                                       0020
      U = (1.0,0.0)                                                    0021
      W = CMPLX(COS(PI/FLOAT(LE1)), SIN(PI/FLOAT(LE1)))                0022
      DO 20 J = 1, LE1                                                 0023
      DO 10 I = J,N,LE                                                 0024
      IP = I + LE                                                      0025
      T = X(IP)*U                                                      0026
      X(IP) = X(I) - T                                                 0027
10    X(I) = X(I) + T                                                  0028
20    U = U*W                                                          0029
      RETURN                                                           0030
      END                                                              0031
```

Resource: Digital Signal Processing
Prof. Alan V. Oppenheim