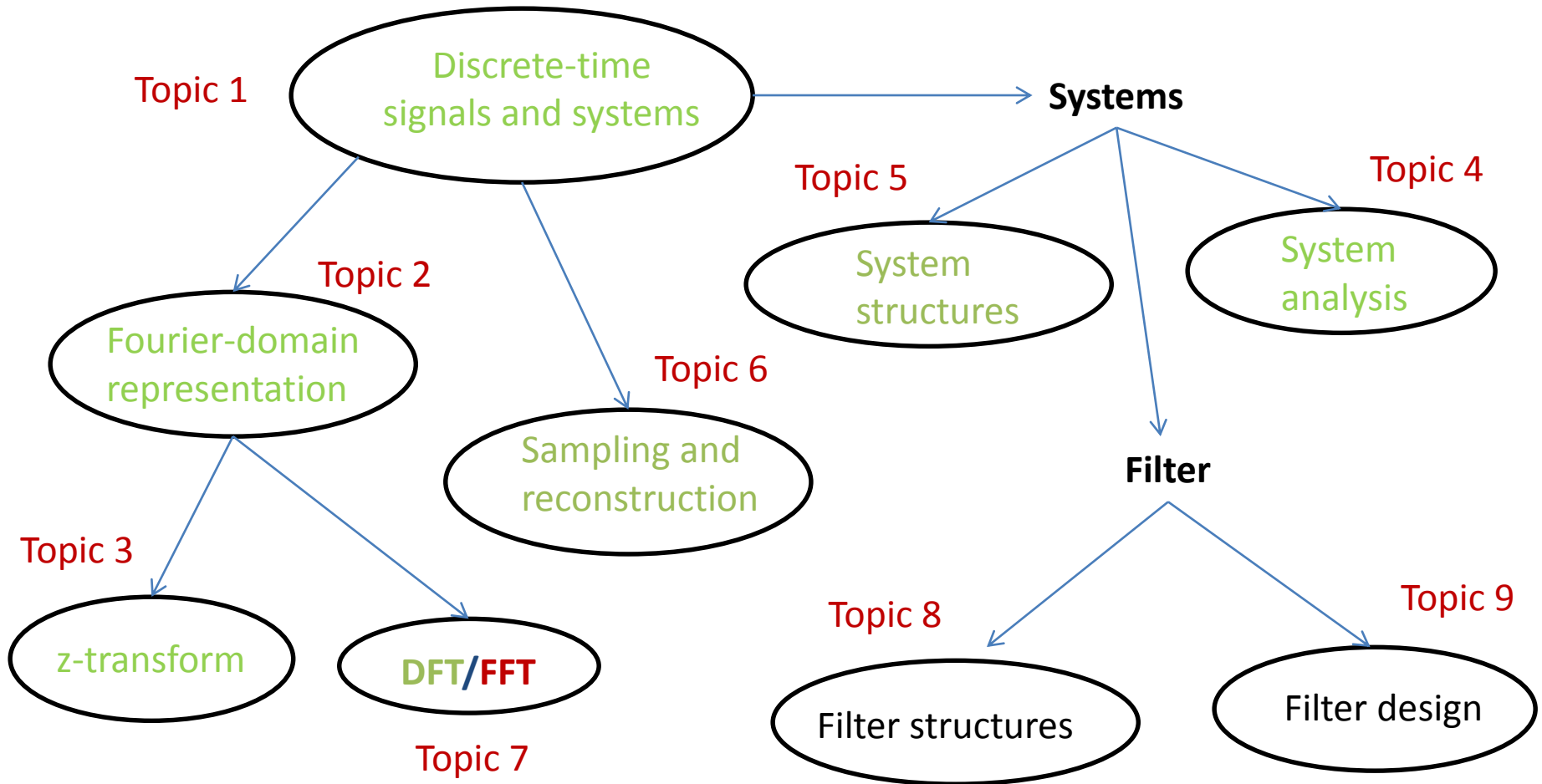


Fast Fourier Transform (FFT)

Chapter 9 in the textbook

Course at a glance



Digital computation of the DFT

- The DFT of a finite-length sequence of length N

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


- The inverse DFT

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

- Due to the duality, focus on the DFT only.
- Use the number of arithmetic multiplications and additions as a measure of computational complexity.
- Fast Fourier transform (FFT) is a set of algorithms for the efficient and digital computation of the N -point DFT, rather than a new transform.

Direct computation of the DFT

- The DFT of a finite-length sequence of length N

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


- Direct computation: N^2 complex multiplications and $N(N-1)$ complex additions

- Compute and store (only over one period)

$$W_N^k = e^{-j(2\pi/N)k}$$

$$= \cos(2\pi k / N) + j \sin(2\pi k / N), \quad k = 0, 1, \dots, N-1$$

- Compute the DFT using stored W_N^k and input $x[n]$

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

W_N^k and $x[n]$ may be complex

Direct computation of the DFT

- For each k

$$X[k] = \sum_{n=0}^{N-1} [(\operatorname{Re}\{x[n]\} \operatorname{Re}\{W_N^{kn}\} - (\operatorname{Im}\{x[n]\} \operatorname{Im}\{W_N^{kn}\}))$$

$$+ j(\operatorname{Re}\{x[n]\} \operatorname{Im}\{W_N^{kn}\} + \operatorname{Im}\{x[n]\} \operatorname{Re}\{W_N^{kn}\})], \quad k = 0, 1, \dots, N-1$$

- Therefore, for each value of k , the direct computation of $X[k]$ requires $4N$ real multiplications and $(4N-2)$ real additions.
- The direct computation of the DFT requires $4N^2$ real multiplications and $N(4N-2)$ real additions.
- The efficiency can be improved by exploiting the symmetry and periodicity properties of W_N^{kn}

Symmetry and periodicity of complex exponential

- Complex conjugate symmetry

$$W_N^{k[N-n]} = W_N^{-kn} = (W_N^{kn})^* = \operatorname{Re}\{W_N^{kn}\} - j \operatorname{Im}\{W_N^{kn}\}$$

- Periodicity in n and k

$$W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n}$$

- For example

$$\begin{aligned} & \operatorname{Re}\{x[n]\} \operatorname{Re}\{W_N^{kn}\} + \operatorname{Re}\{x[N-n]\} \operatorname{Re}\{W_N^{k[N-n]}\} \\ &= (\operatorname{Re}\{x[n]\} + \operatorname{Re}\{x[N-n]\}) \operatorname{Re}\{W_N^{kn}\} \end{aligned}$$

- The number of multiplications is reduced by a factor of 2.

FFT

- Cooley and Tukey (1965) published an algorithm for the computation of the DFT that is applicable when N is a composite number, i.e., the product of two or more integers. Later, it resulted in a number of highly efficient computational algorithms.
- The entire set of such algorithms are called the fast Fourier transform, FFT.
- FFT decomposes the computation of the DFT of a sequence of length N into successively smaller DFTs.

Decimation-in-time FFT algorithms

- Where
 - decomposition is done by decomposing the sequence into successively smaller subsequences,
 - and both the symmetry and periodicity of complex exponential $W_N^{kn} = e^{-j(2\pi/N)kn}$ are exploited.
- Consider $N = 2^v$ and separate $x[n]$ into two $(N/2)$ -point sequences

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

$$X[k] = \sum_{n \text{ even}} x[n]W_N^{kn} + \sum_{n \text{ odd}} x[n]W_N^{kn}$$

Decimation-in-time FFT algorithms

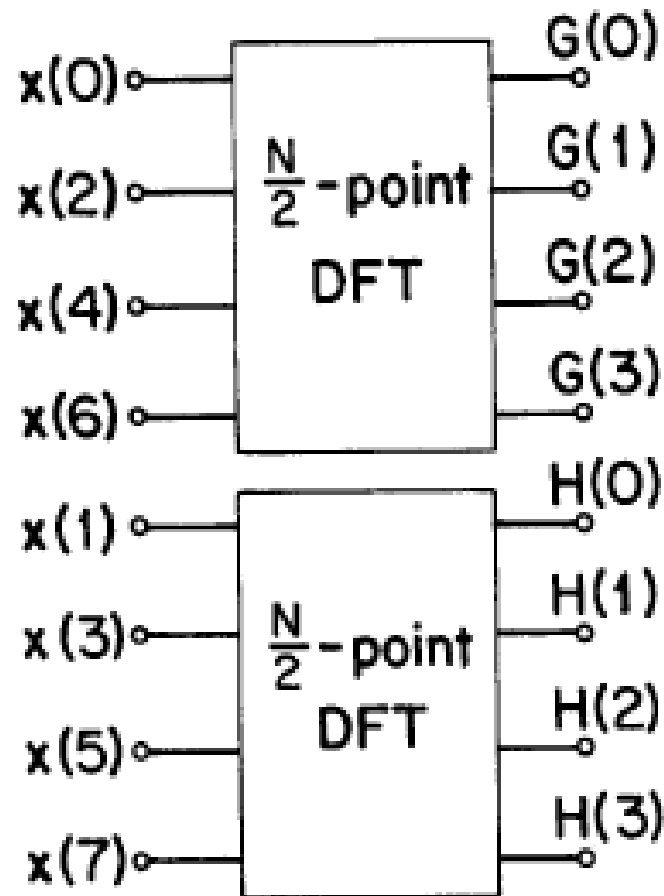
$$\begin{aligned} X[k] &= \sum_{n \text{ even}} x[n]W_N^{kn} + \sum_{n \text{ odd}} x[n]W_N^{kn} \\ &= \sum_{r=0}^{(N/2)-1} x[2r]W_N^{2rk} + \sum_{r=0}^{(N/2)-1} x[2r+1]W_N^{(2r+1)k} \\ &= \sum_{r=0}^{(N/2)-1} x[2r](W_N^2)^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1](W_N^2)^{rk} \\ &= \sum_{r=0}^{(N/2)-1} x[2r]W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} x[2r+1]W_{N/2}^{rk} \\ &= \underline{G[k] + W_N^k H[k]}, \quad k = 0, 1, \dots, N-1 \end{aligned}$$

(only compute for $k = 0, 1, \dots, N/2 - 1$) due to the periodicity

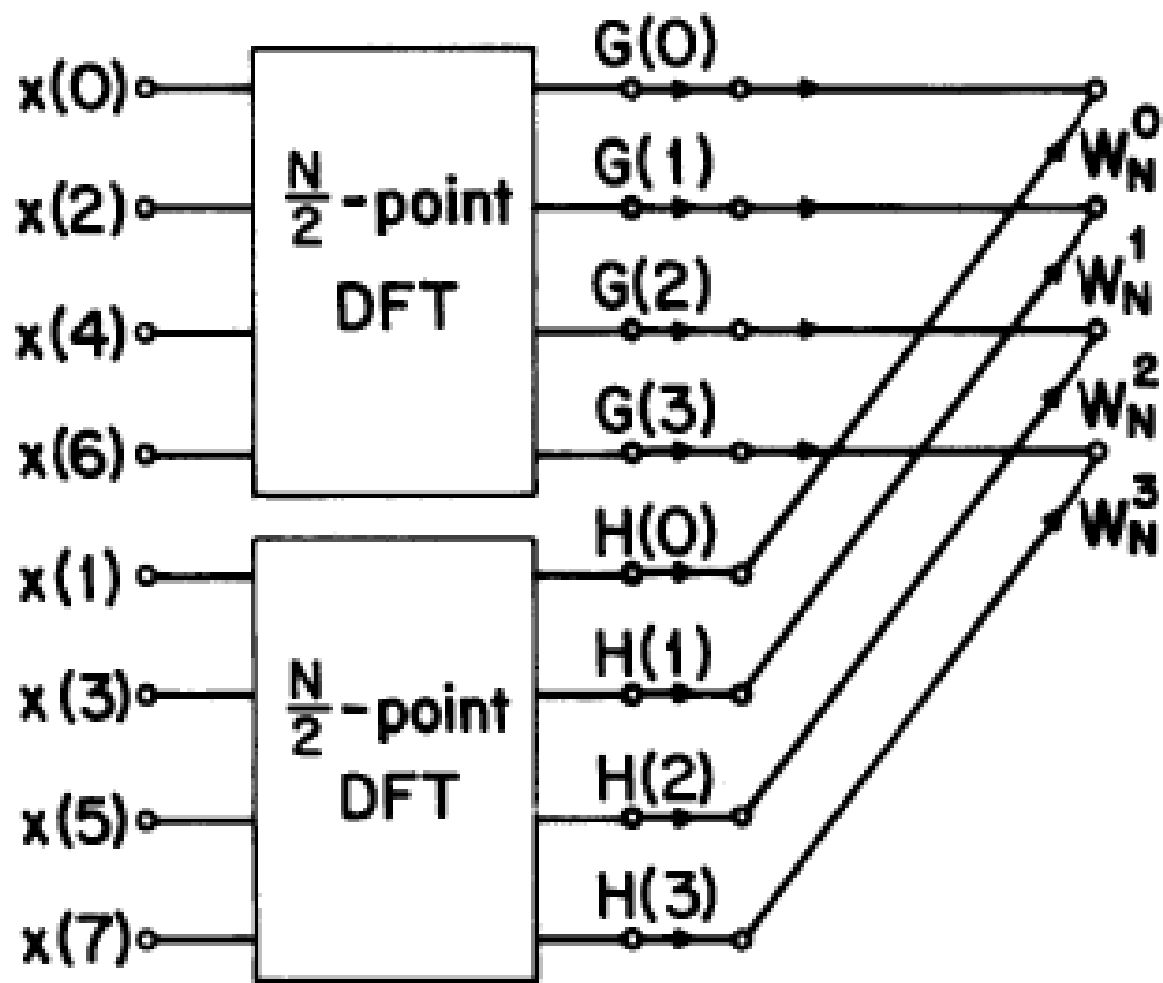
$$\begin{aligned}
 \underline{X}(k) = & \underbrace{\sum_{r=0}^{\frac{N}{2}-1} X(2r) W_{N/2}^{rk}}_{\substack{\frac{N}{2} \text{ POINT DFT} \\ G(k)}} + W_N^k \underbrace{\sum_{r=0}^{\frac{N}{2}-1} X(2r+1) W_{N/2}^{rk}}_{\substack{\frac{N}{2} \text{ POINT DFT} \\ H(k)}}
 \end{aligned}$$

$$\underline{X}(k) = G(k) + W_N^k H(k)$$

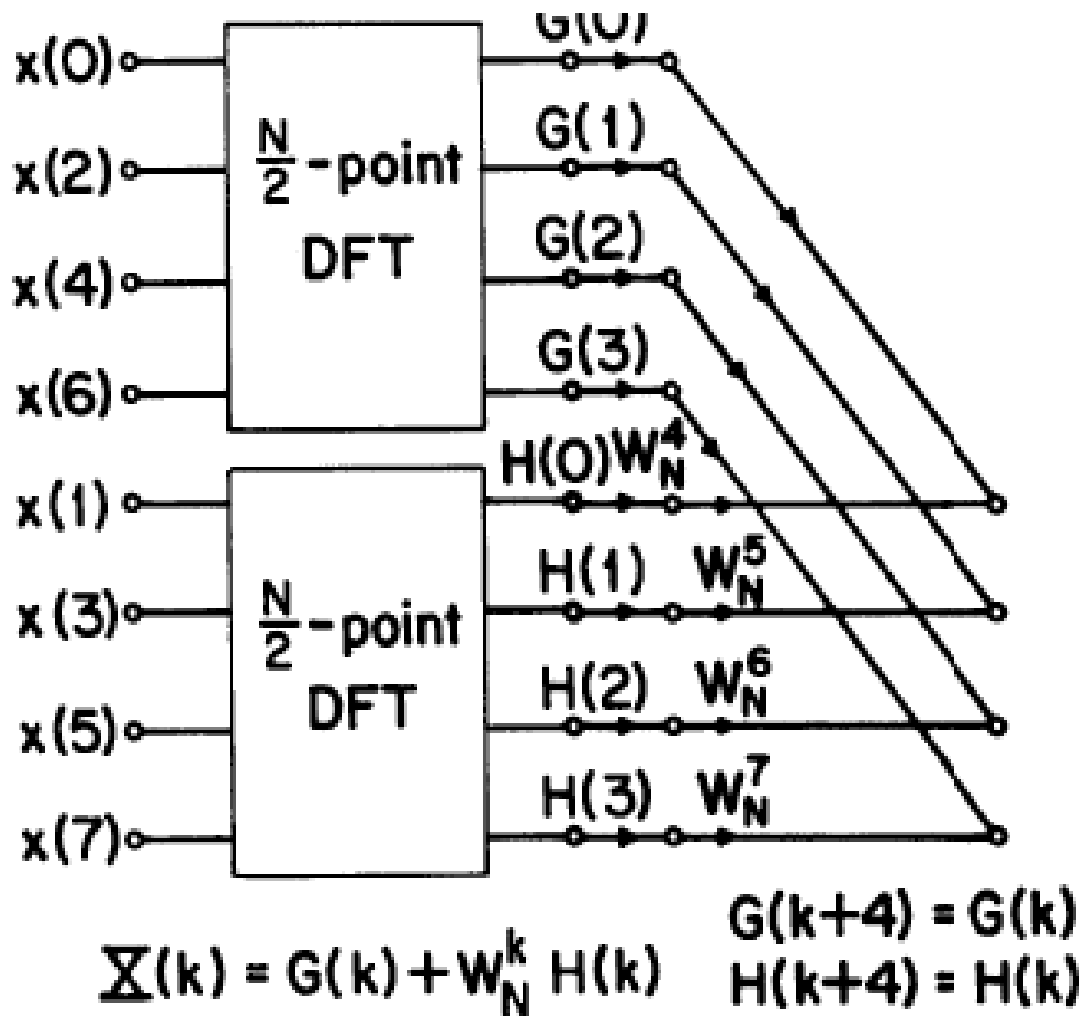
$$2\left(\frac{N}{2}\right)^2 + N = N + \frac{N^2}{2}$$



$$\Sigma(k) = G(k) + W_N^k H(k)$$



$$\Sigma(k) = G(k) + W_N^k H(k)$$



Flow graph of the decimation-in-time

- Periodicity is applied, e.g. $G[7]=G[3]$

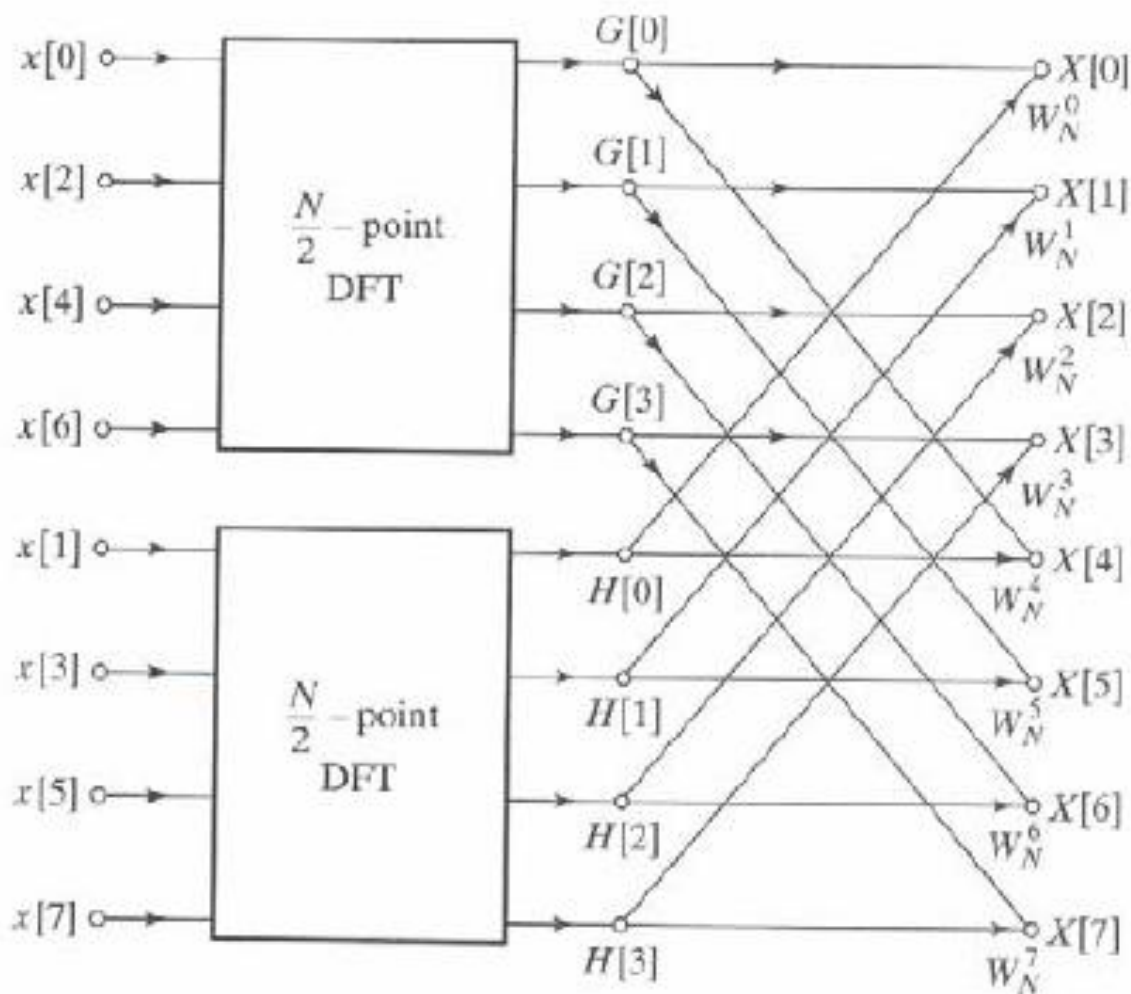


Figure 9.3 Flow graph of the decimation-in-time decomposition of an N -point DFT computation into two $(N/2)$ -point DFT computations ($N = 8$).

Decimation-in-time FFT

- Further break down

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

$$= \sum_{l=0}^{(N/4)-1} g[2l] W_{N/4}^{lk} + W_{N/2}^k \sum_{l=0}^{(N/4)-1} g[2l+1] W_{N/4}^{lk}$$

$$H[k] = \sum_{l=0}^{(N/4)-1} h[2l] W_{N/4}^{lk} + W_{N/2}^k \sum_{l=0}^{(N/4)-1} h[2l+1] W_{N/4}^{lk}$$

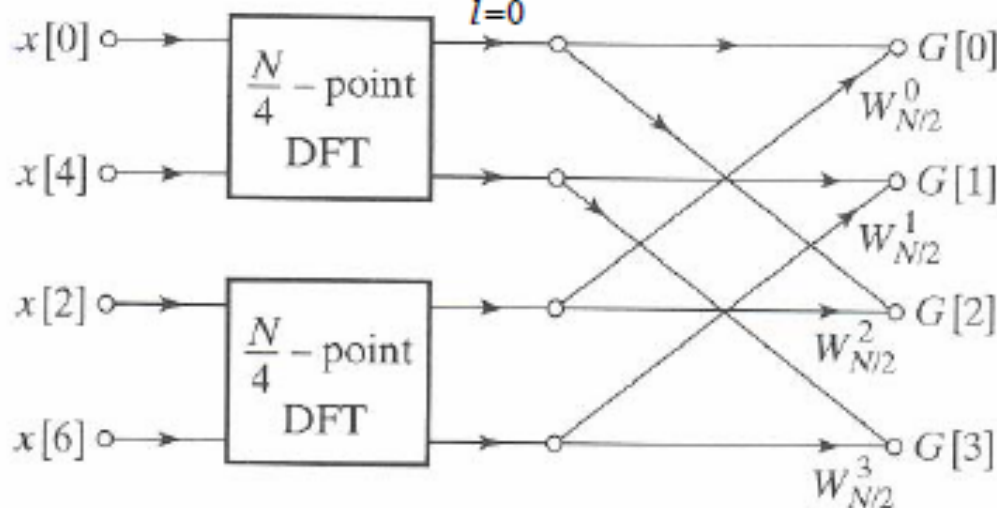


Figure 9.4 Flow graph of the decimation-in-time decomposition of an $(N/2)$ -point DFT computation into two $(N/4)$ -point DFT computations ($N = 8$).

Combination of Fig. 9.3 and 9.4

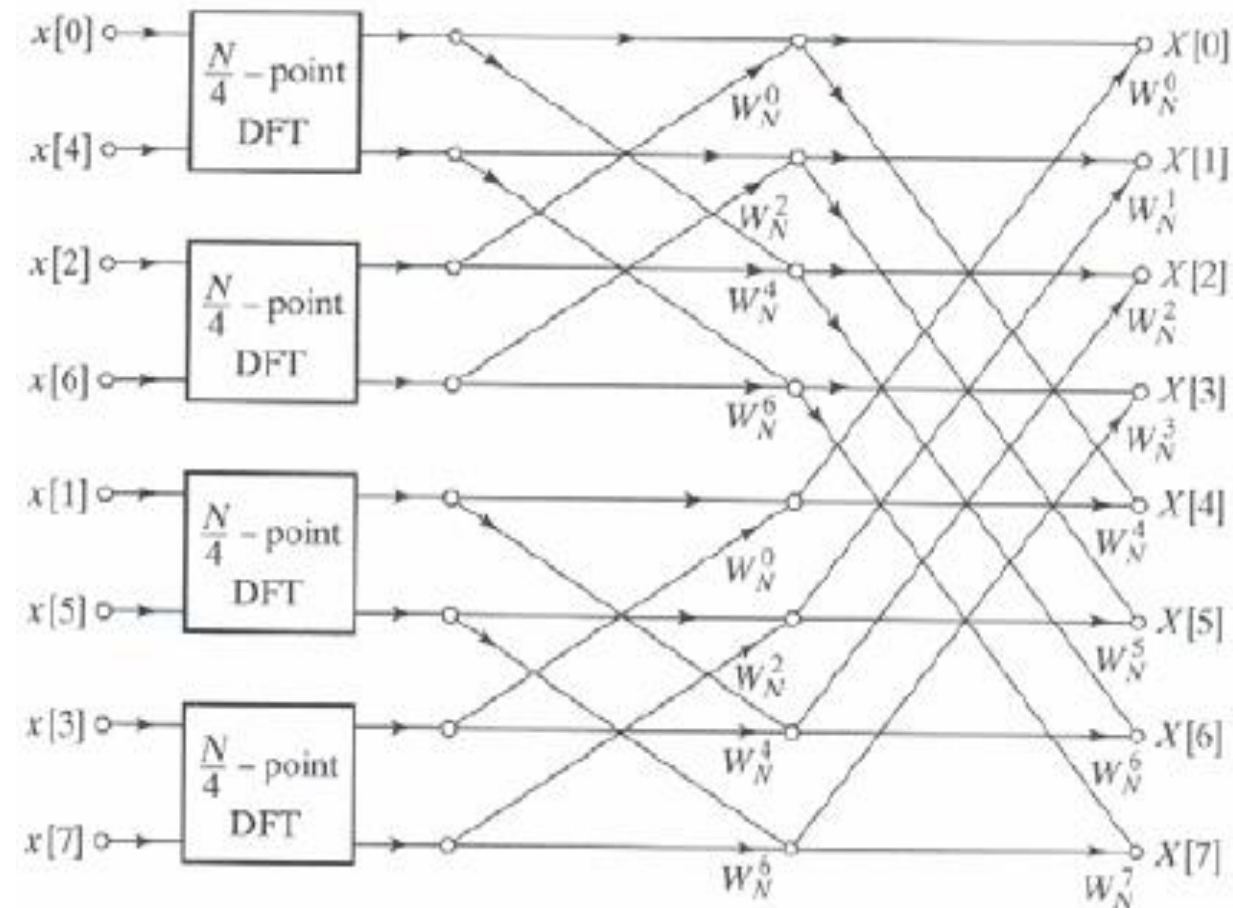


Figure 9.5 Result of substituting the structure of Figure 9.4 into Figure 9.3.

2-point DFT

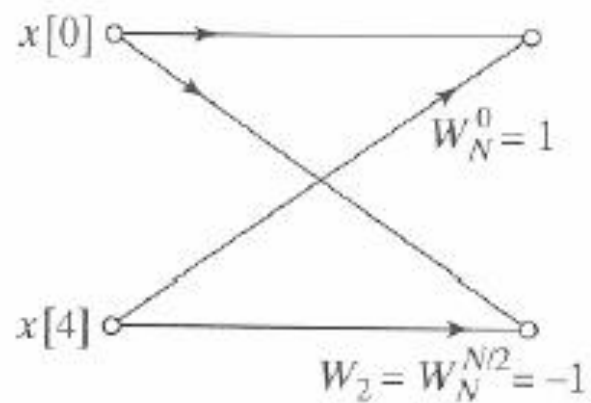
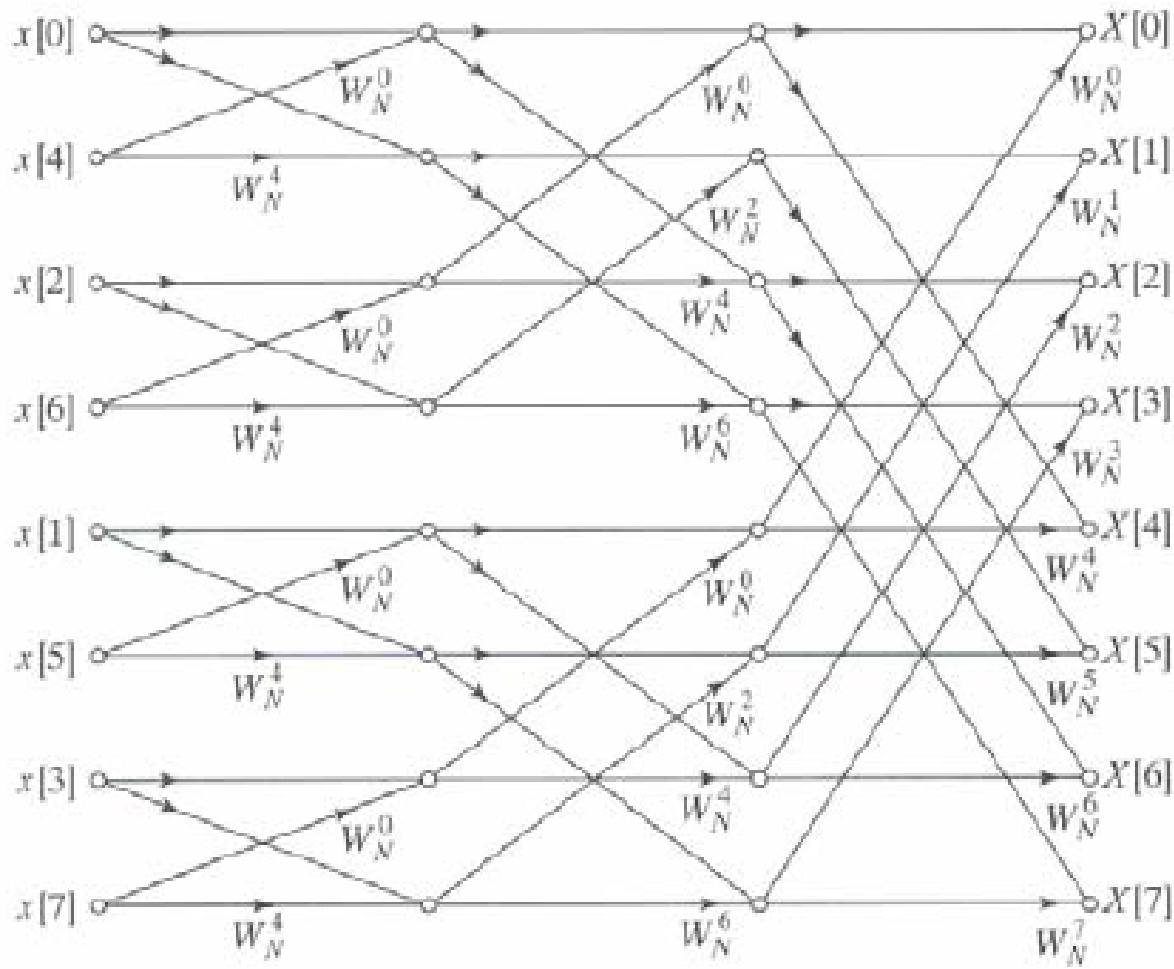


Figure 9.6 Flow graph of a 2-point DFT.

Flow graph



$\log_2 N$ stages and
each stage has N
complex multiplications
and N complex
additions .

In total, $N \log_2 N$ complex
multiplications and additions

e.g.

$$N = 2^{10} = 1024$$

$$N^2 = 1,048,576$$

$$N \log_2 N = 10,240$$

A reduction of 2 orders!

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

$$N = 2^{\nu}$$

1 N POINT DFT

2 $\frac{N}{2}$ POINT DFT'S +

$$N^2 \downarrow$$

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

$$\left(\frac{N}{2}\right)^2 \rightarrow 2\left(\frac{N}{4}\right)^2 + \frac{N}{2}$$

4 $\frac{N}{4}$ POINT DFT'S +

$$4\left(\frac{N}{4}\right)^2 + N + N$$

$$\left(\frac{N}{4}\right)^2 \rightarrow 2\left(\frac{N}{8}\right)^2 + \frac{N}{4}$$

8 $\frac{N}{8}$ POINT DFT'S +

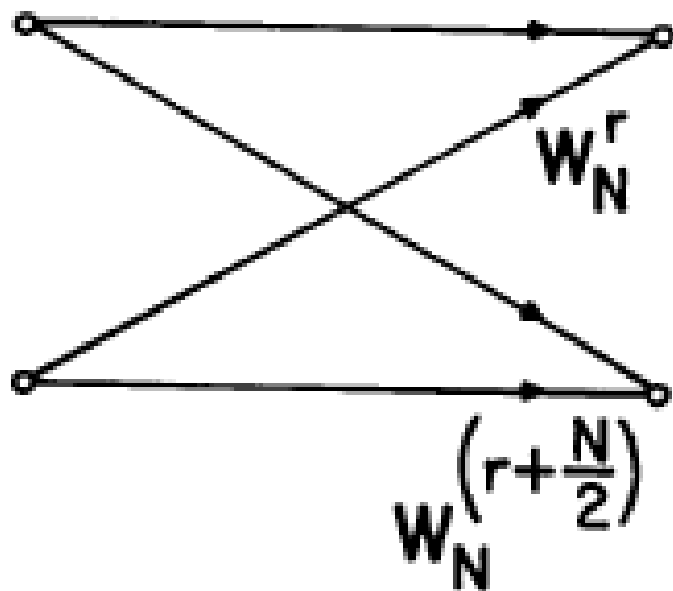
$$8\left(\frac{N}{8}\right)^2 + N + N + N$$

$$\left(\frac{N}{8}\right)^2 \rightarrow 2\left(\frac{N}{16}\right)^2 + \frac{N}{8}$$

⋮

$$\underbrace{N + N + \dots + N}_{\nu \text{ TIMES}}$$

ν TIMES



$$W_N^{(r + \frac{N}{2})} = W_N^r W_N^{\frac{N}{2}}$$

$$W_N^{\frac{N}{2}} = e^{-j \frac{2\pi}{N} \frac{N}{2}} = e^{-j\pi} = -1$$

Flow graph of butterfly computation

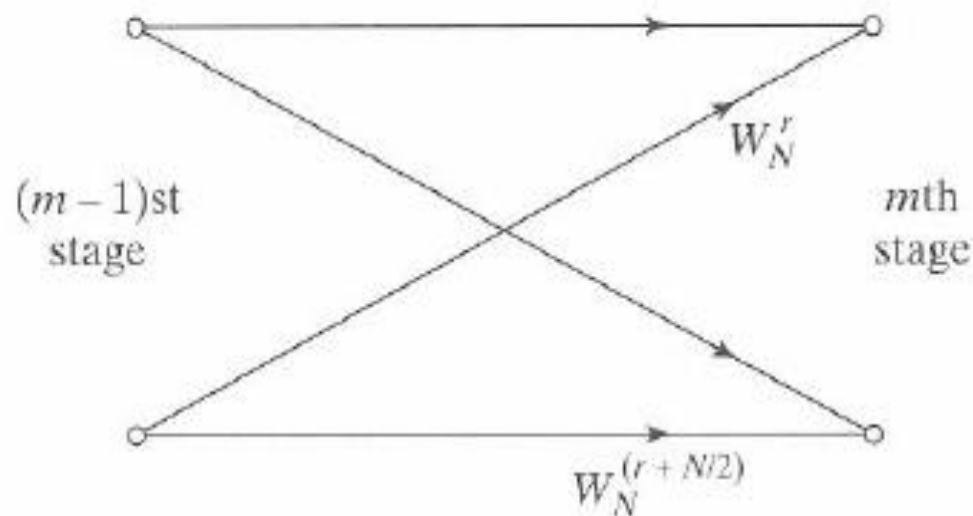


Figure 9.8 Flow graph of basic butterfly computation in Figure 9.7.

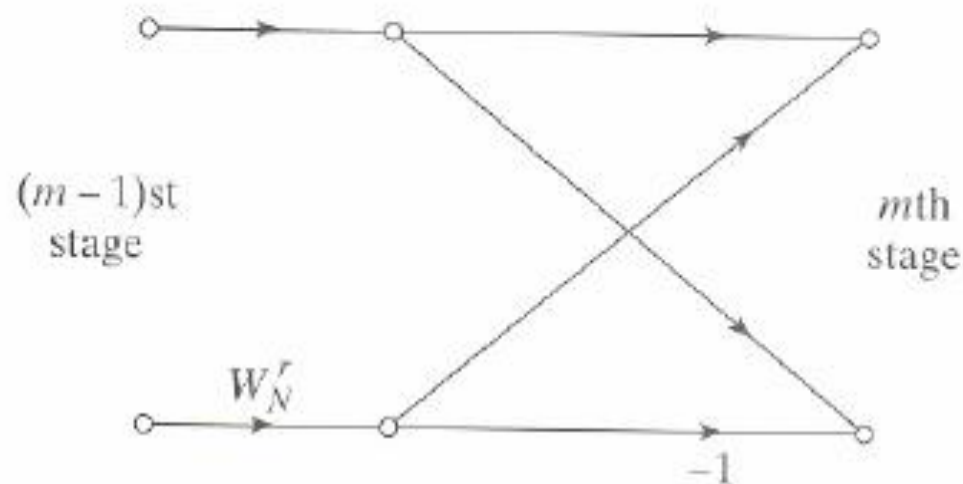


Figure 9.9 Flow graph of simplified butterfly computation requiring only one complex multiplication.

Flow graph

- The number of complex multiplications are reduced by a factor of 2 over the number in Fig. 9.7.

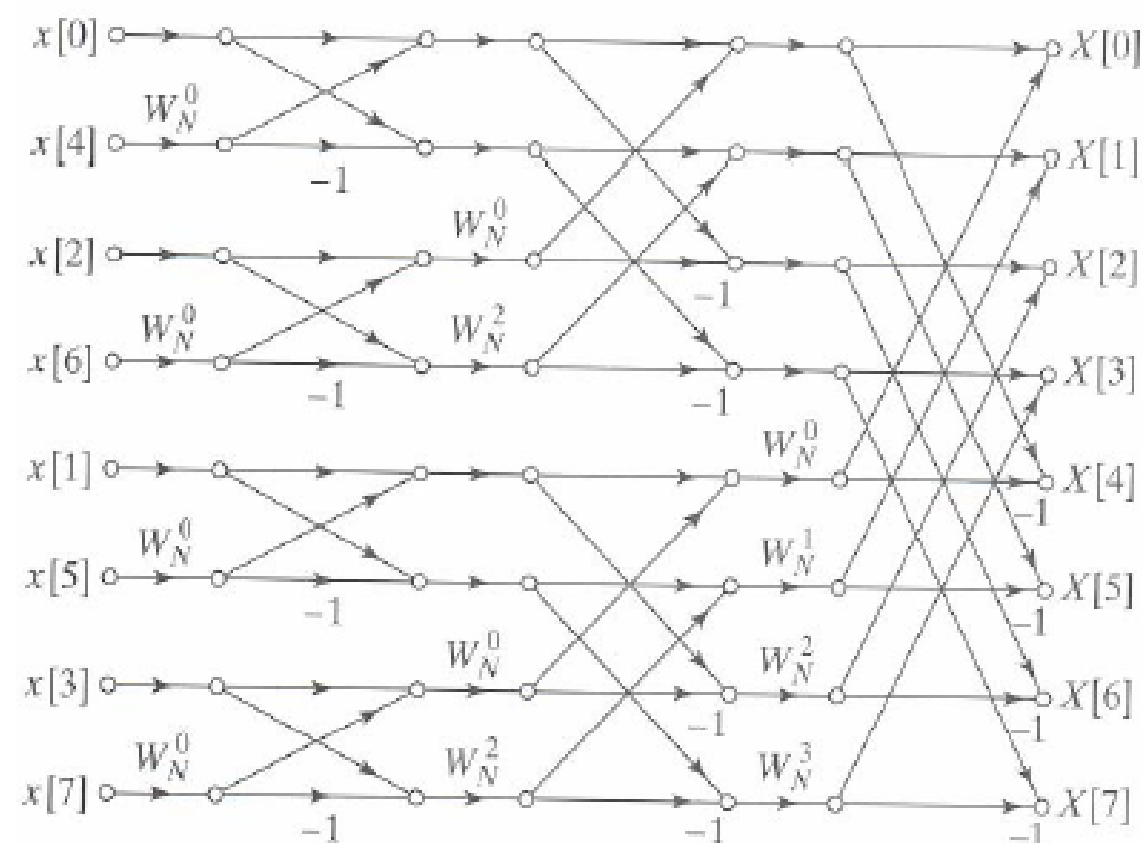


Figure 9.10 Flow graph of 8-point DFT using the butterfly computation of Figure 9.9.

In-place computation

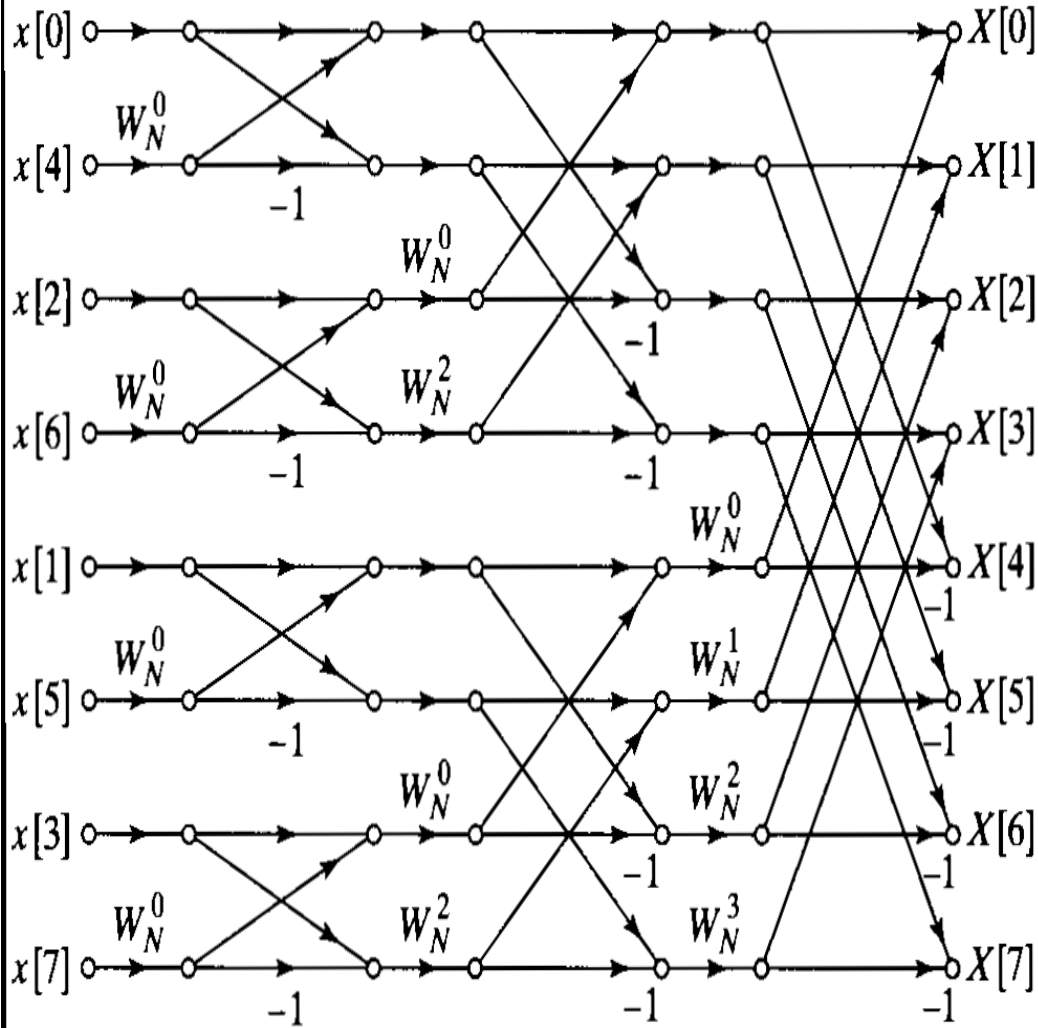


Figure 9.10 Flow graph of 8-point DFT using the butterfly computation of Figure 9.9.

Normal order input – Bit-reversed output

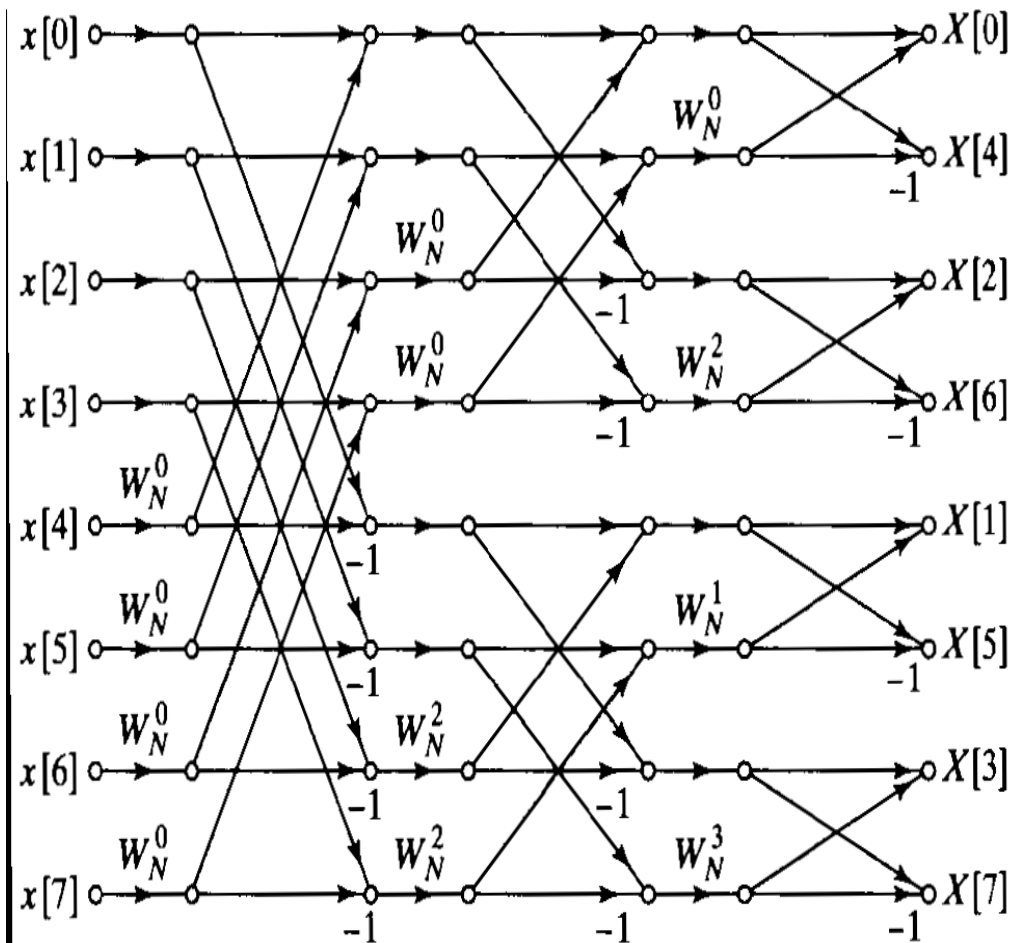


Figure 9.14 Rearrangement of Figure 9.10 with input in normal order and output in bit-reversed order.

Normal order input – normal order output

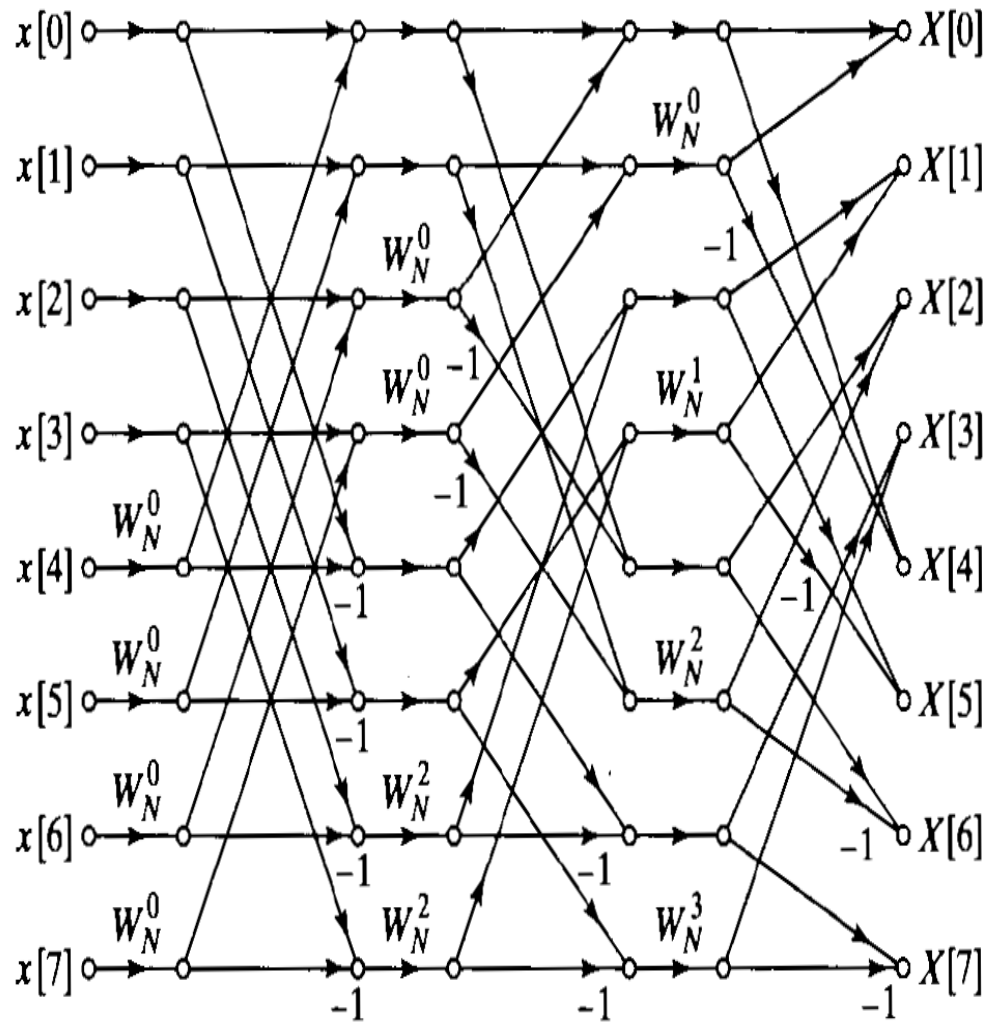


Figure 9.15 Rearrangement of Figure 9.10 with both input and output in normal order.

Sequential memory

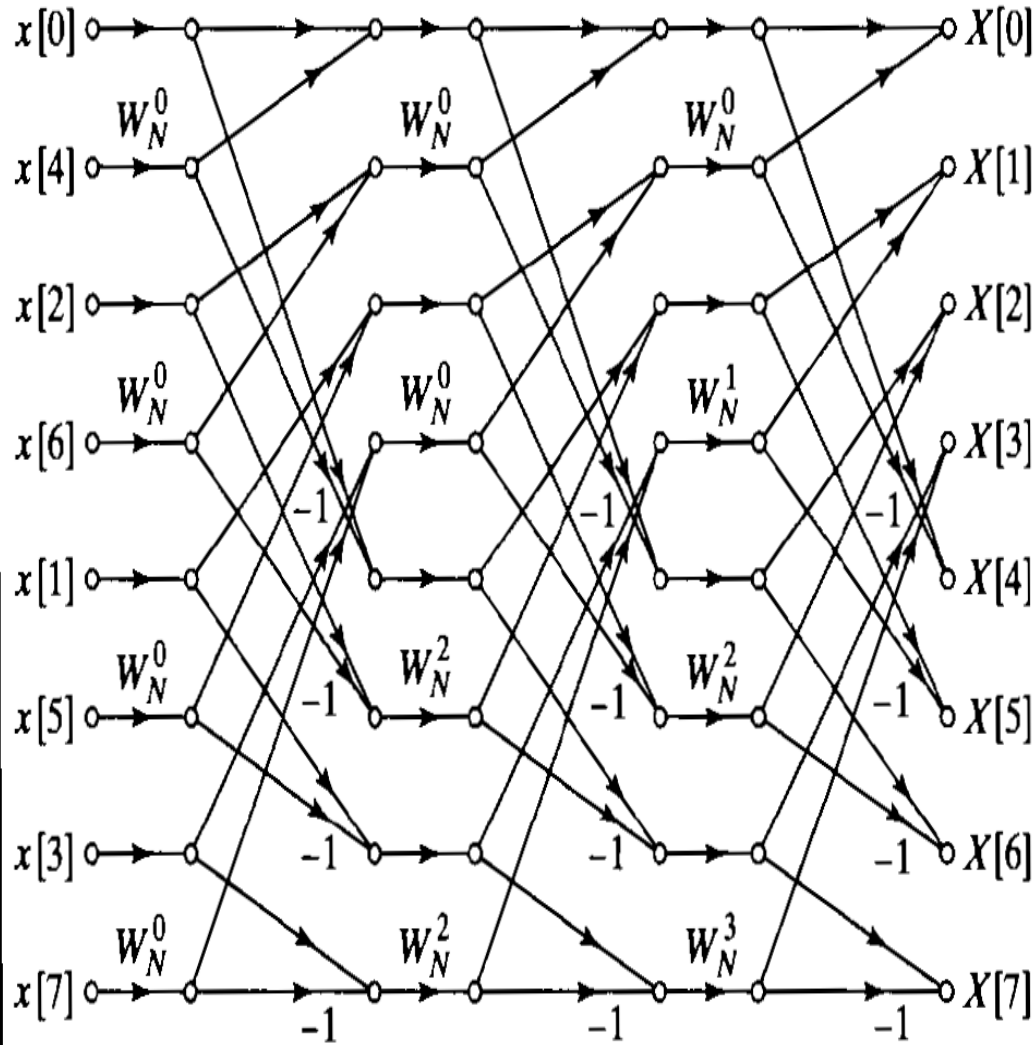
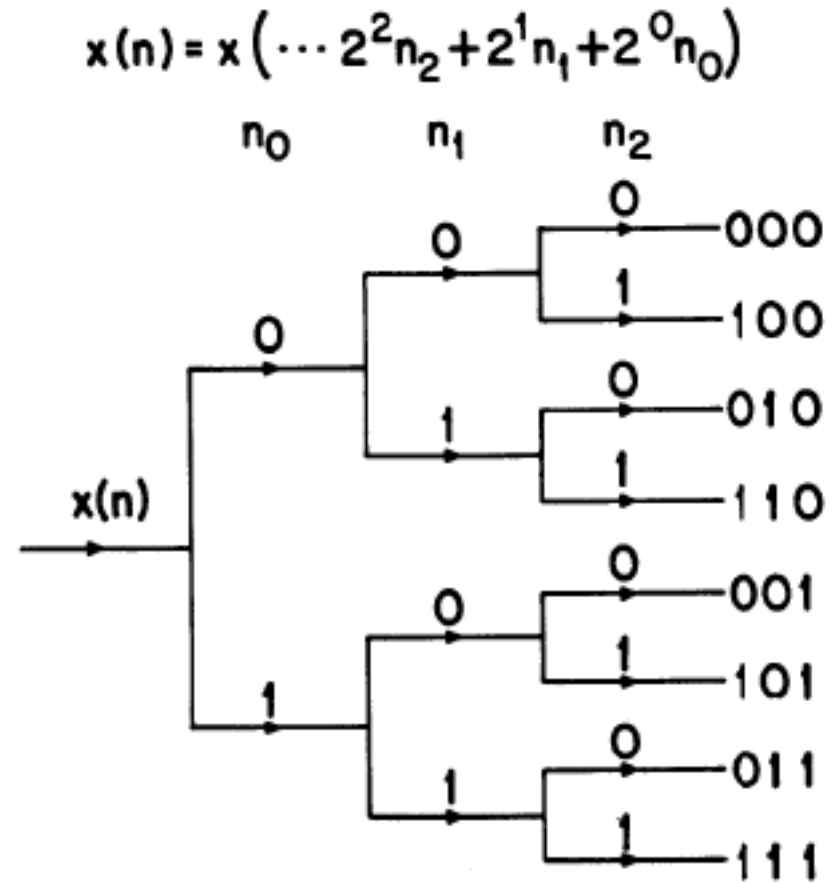


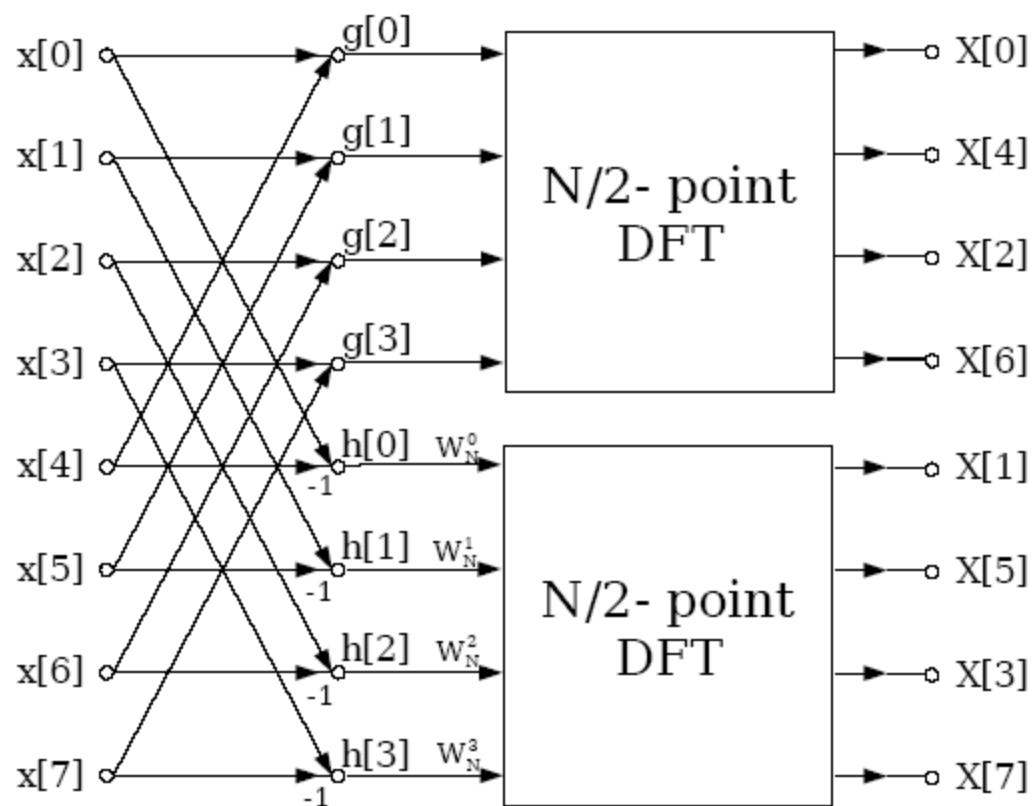
Figure 9.16 Rearrangement of Figure 9.10 having the same geometry for each stage, thereby permitting sequential data accessing and storage.

Bit-reverse order

	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



Decimation-in-frequency FFT algorithms



Decimation – in – frequency FFT

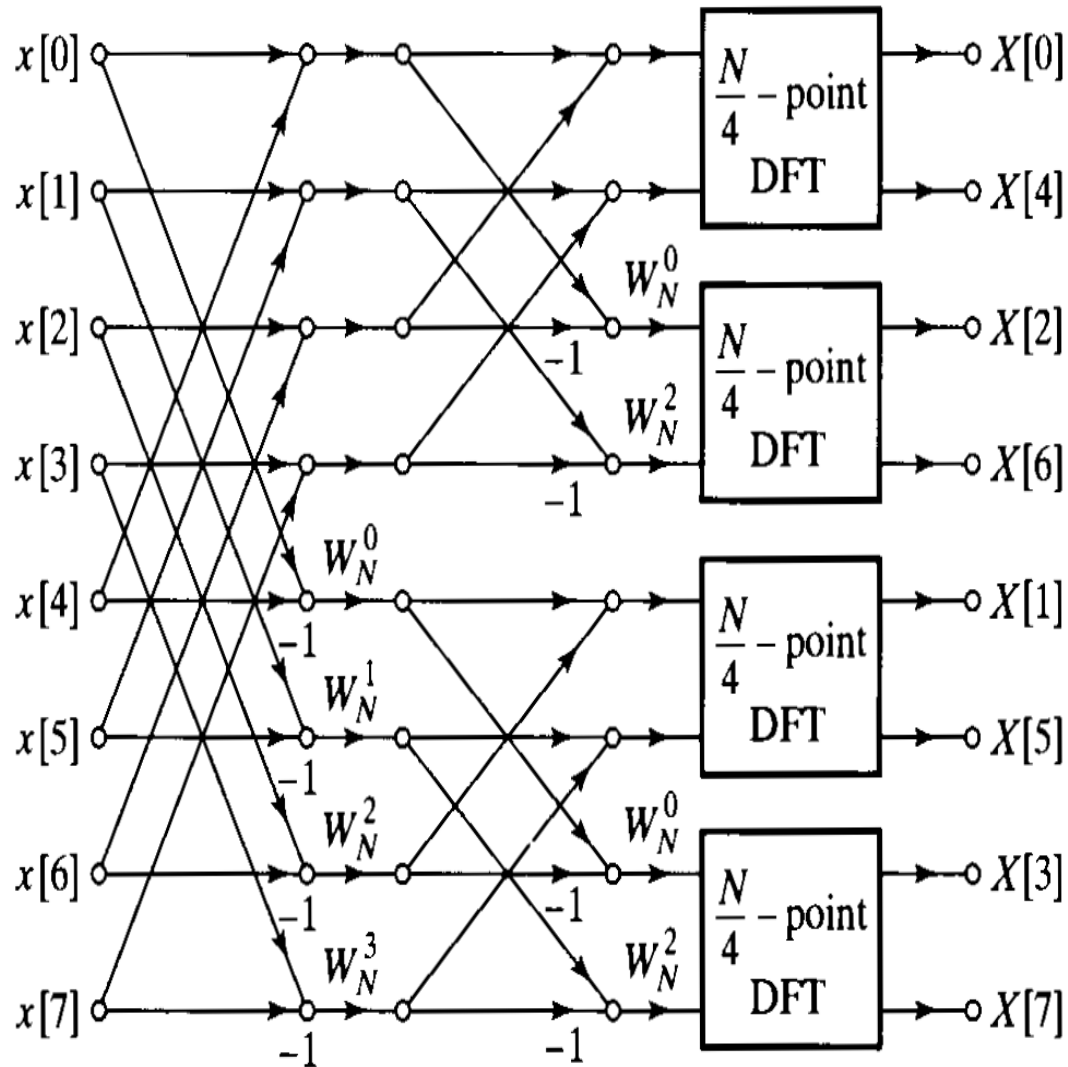


Figure 9.18 Flow graph of decimation-in-frequency decomposition of an 8-point DFT into four 2-point DFT computations.

2-point DFT for decimation-in-frequency

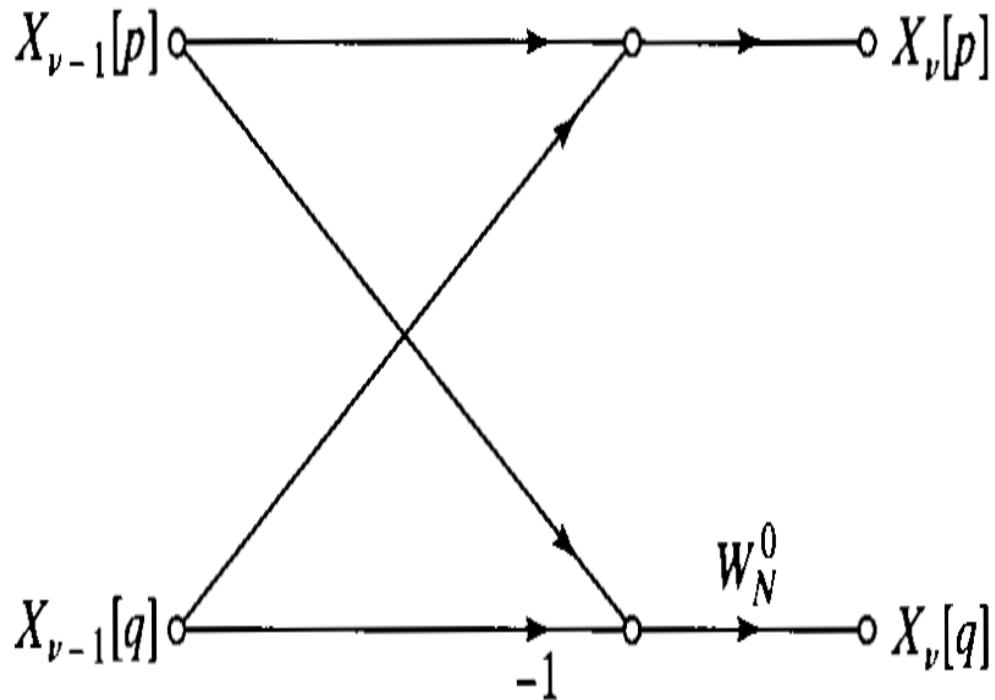


Figure 9.19 Flow graph of a typical 2-point DFT as required in the last stage of decimation-in-frequency decomposition.

Flow graph of DIF of an 8-point DFT

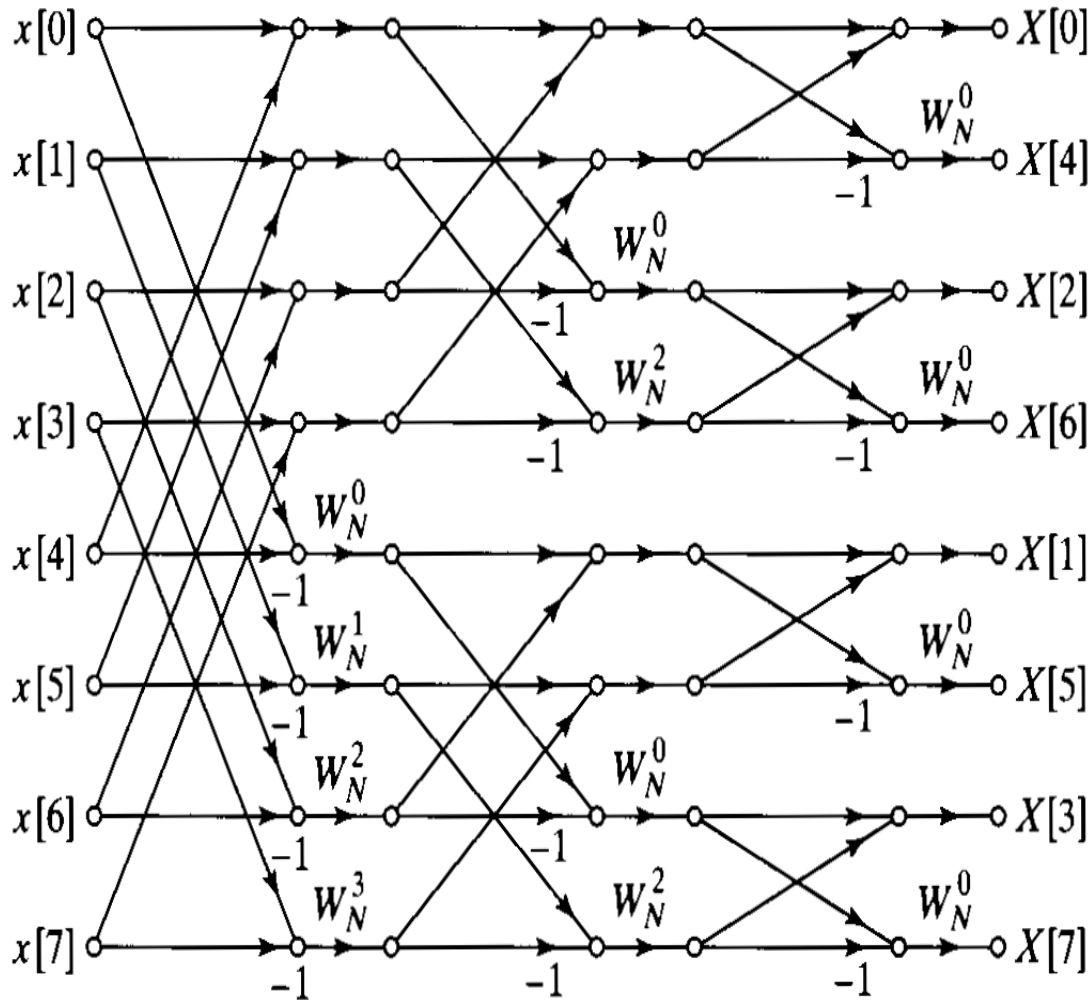


Figure 9.20 Flow graph of complete decimation-in-frequency decomposition of an 8-point DFT computation.

Typical Butterfly for DIF FFT

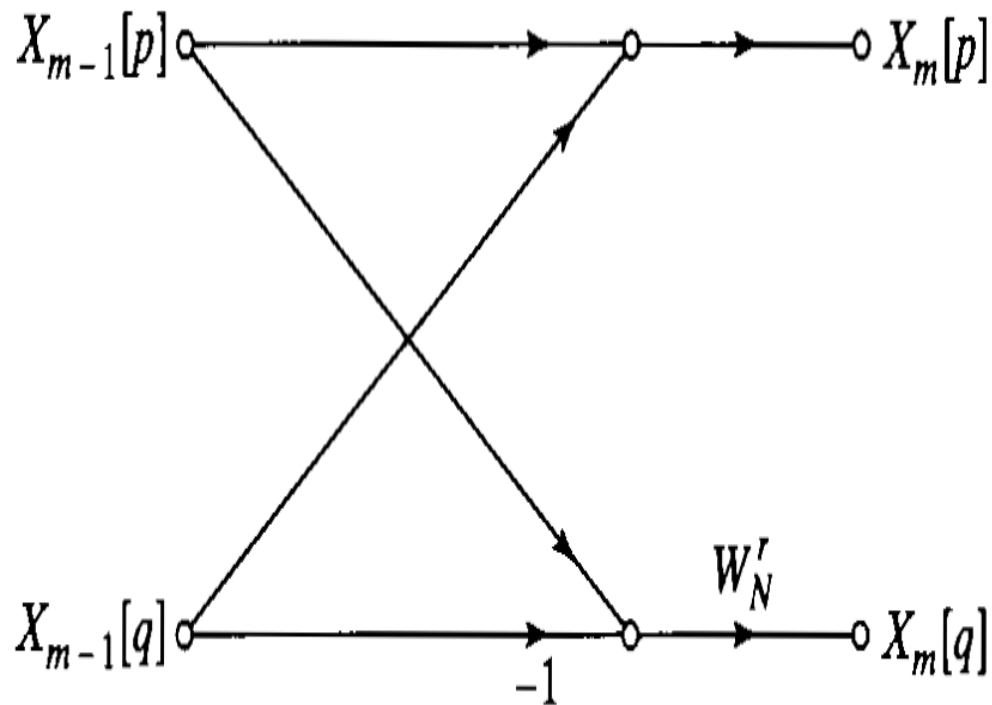


Figure 9.21 Flow graph of a typical butterfly computation required in Figure 9.20.

Normal order output – bit-reversed order input

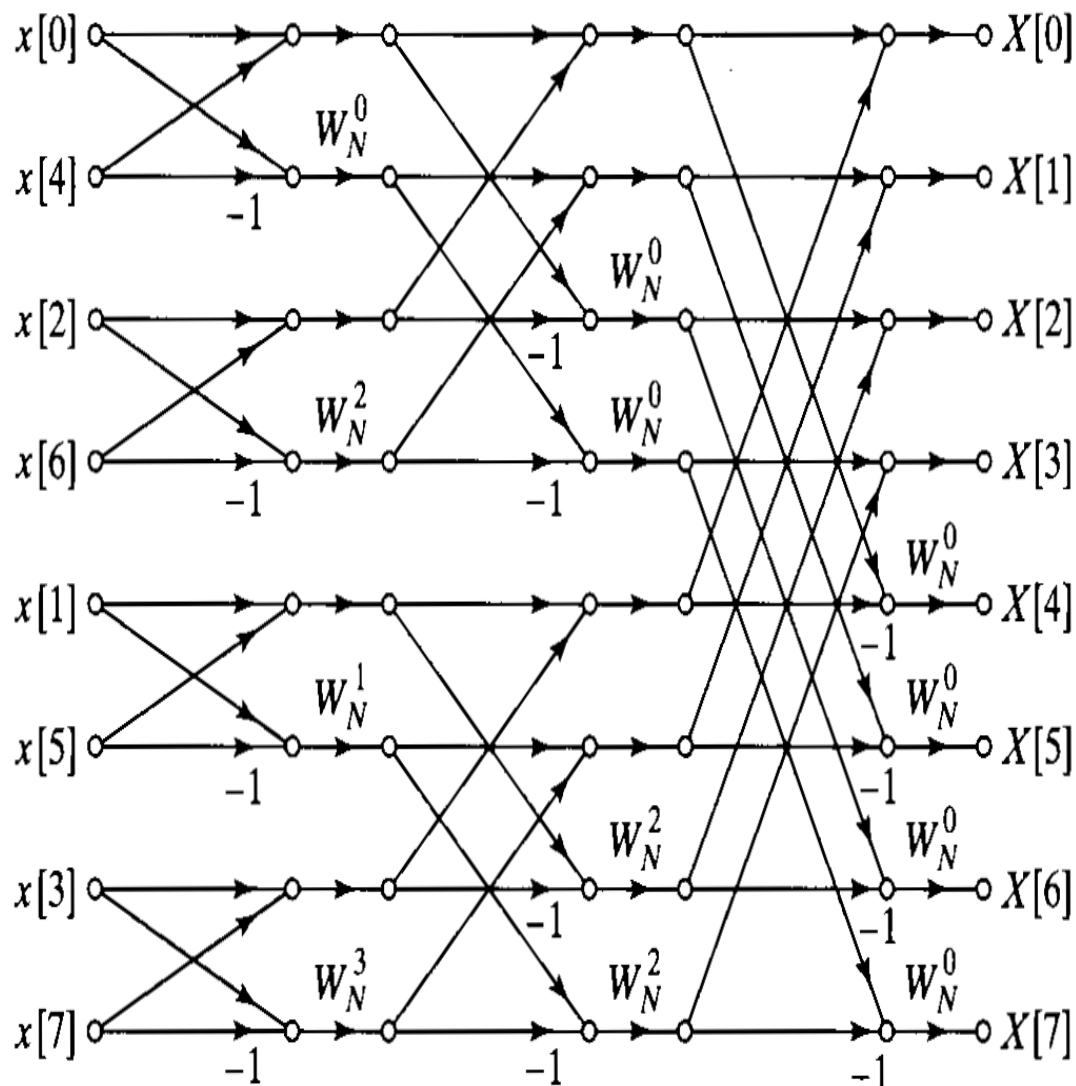


Figure 9.22 Flow graph of a decimation-in-frequency DFT algorithm obtained from Figure 9.20. Input in bit-reversed order and output in normal order. (Transpose of Figure 9.14.)

Normal order output – normal order input

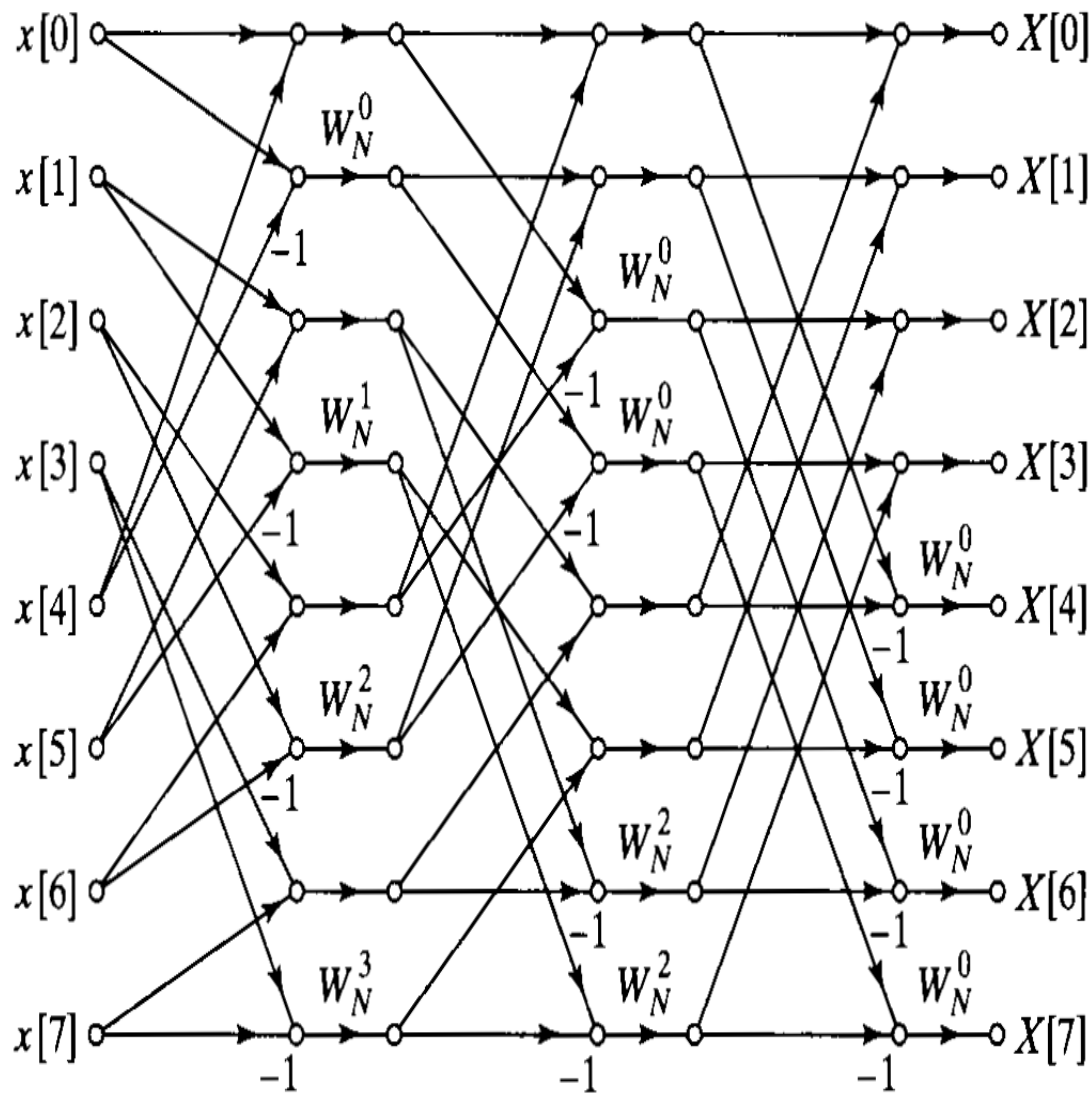


Figure 9.23 Rearrangement of Figure 9.20 with both input and output in normal order. (Transpose of Figure 9.15.)

Sequential memory

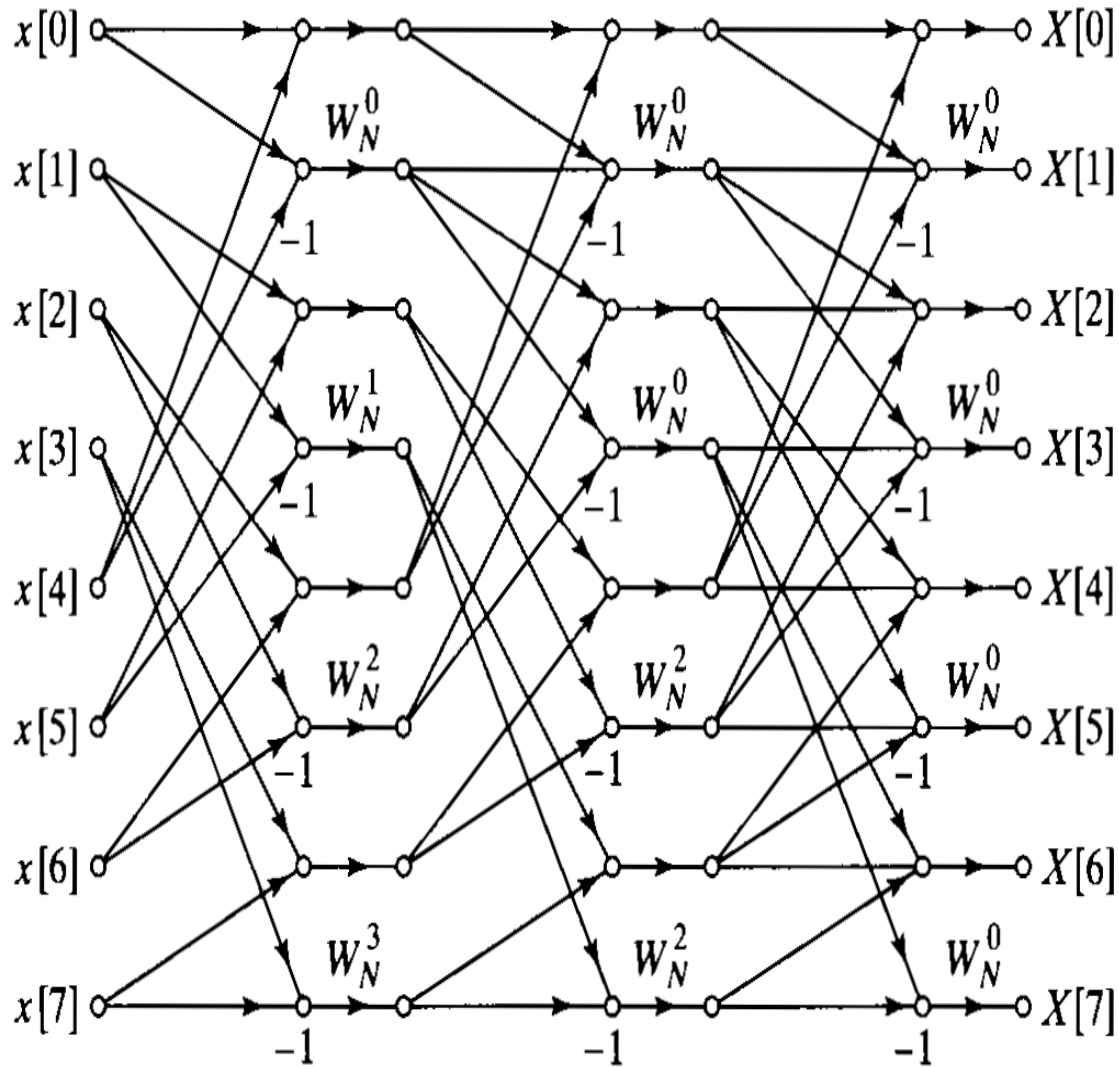


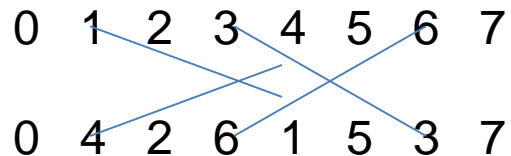
Figure 9.24 Rearrangement of Figure 9.20 having the same geometry for each stage, thereby permitting sequential data accessing and storage. (Transpose of Figure 9.16.)

Some computational considerations

1. Inverse DFT:

- $1/N$ scaling
- conjugate powers of W (coefficients) or,
- flipping input $x(-n)$

2. Bit-reversal (in-place computation)



Swap place 1 , 4

Swap place 3, 6

3. Dealing with coefficients:

- store them in a table
- generate them as they are needed

In FFT DIF $W \rightarrow$ normal order

In FFT DIT $W \rightarrow$ bit-reversed order

FFT for convolution / correlation

For avoiding bit-reversed order :

- DFT – choose algorithm where input is in normal order and output is in bit-reversed order.
- Store impulse response in bit-reversed order
- Do multiplication
- IDFT – choose an FFT algorithm where input is in bit-reversed order and output in normal order

DFT: decimation-in-time (normally order input, bit-reversed output)

* Coefficients W_N are in bit-reversed order

IDFT: decimation-in-frequency (bit-reversed input, normally order output)

* Coefficients W_N are in normal order

To match up coefficients order:

-Decimation-in-frequency (normally input, bit-reversed output)

* Coefficients are in normal order

-Decimation-in-time (bit-reversed input, normally output)

*Coefficients are in normal order

General Radix FFT

- All of the previous algorithms are Radix-2 FFT Algorithms ,i.e. $N=2^v$
- N can be any integer number – general Radix FFT algorithms