Chapter 8

Efficient Computation of the DFT: Fast Fourier Transform Algorithms

清大電機系林嘉文
cwlin@ee.nthu.edu.tw
03-5731152

8.1 Efficient computation of the DFT

- DFT formulation

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

where $W_N = e^{-j2\pi/N}$

- The IDFT formulation

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

- How to compute the DFT efficiently?

8.1 Efficient computation of the DFT

- Direct computation of the DFT is basically inefficient, primarily because it does not exploit the symmetry and periodicity of the phase factor W_N

$$\text{Symmetry property: } W_N^{k+N/2} = -W_N^k$$
$$\text{Periodicity property: } W_N^{k+N} = W_N^k$$

8.1.1 Direct computation of the DFT

- For a complex-valued sequence $x(n)$ of N points, the DFT may be expresses as

$$X_R(k) = \sum_{n=0}^{N-1}[x_R(n)\cos\frac{2\pi kn}{N} + x_I(n)\sin\frac{2\pi kn}{N}]$$
$$X_R(k) = -\sum_{n=0}^{N-1}[x_R(n)\sin\frac{2\pi kn}{N} - x_I(n)\cos\frac{2\pi kn}{N}]$$

- The direct computation requires:

  $2N^2$ evaluations of trigonometric functions

  $4N^2$ real multiplications

  $4N(N-1)$ real additions

  A number of indexing and addressing operations

8.1.2 Divide-and-Conquer Approach

- Decomposition of an N-point DFT into successively smaller DFTs ➔ $N=LM$

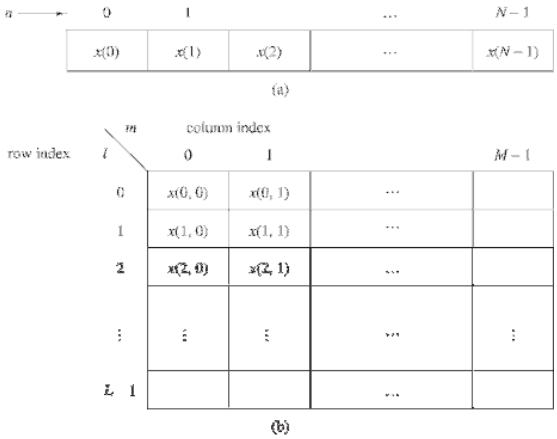


Figure 8.1.1 Two dimensional data array for storing the sequence $x(n)$, $0 \le n \le N-1$.

8.1.2 Divide-and-Conquer Approach

- Mapping function
  - row-wise mapping ➔ $n = Ml + m \quad 0 \le p \le L\text{-}1$, $k = Mp + q \quad 0 \le q \le M\text{-}1$
  - column-wise mapping ➔ $n = l + mL$, $k = p + qL$

Column-wise $n = l + mL$

l \ m	0	1	2	...	$M-1$
0	$x(0)$	$x(L)$	$x(2L)$	...	$x((M-1)L)$
1	$x(1)$	$x(L+1)$	$x(2L+1)$	...	$x((M-1)L+1)$
2	$x(2)$	$x(L+2)$	$x(2L+2)$	...	$x((M-1)L+2)$
⋮	⋮	⋮	⋮	⋯	⋮
$L-1$	$x(L-1)$	$x(2L-1)$	$x(3L-1)$	...	$x(LM-1)$

(b)

8.1.2 Divide-and-Conquer Approach

$$X(p,q)=\sum_{m=0}^{M-1}\sum_{l=0}^{L-1}x(l,m)W_N^{(Mp+q)(mL+l)} \qquad X(k)=X(p,q)$$
$$x(n)=x(l,m)$$

but

$$W_N^{(Mp+q)(mL+l)}=W_N^{MLmp}W_N^{mLq}W_N^{Mpl}W_N^{lq}$$

where

$$W_N^{Nmp}=1,\ W_N^{mqL}=W_{N/L}^{mq}=W_M^{mq},\ W_N^{Mpl}=W_{N/M}^{pl}=W_L^{pl}$$

With these simplications, it can be expressed as

$$X(p,q)=\sum_{l=0}^{L-1}\left\{W_N^{lq}\left[\sum_{m=0}^{M-1}x(l,m)W_M^{mq}\right]\right\}W_L^{lp}$$

8.1.2 Divide-and-Conquer Approach

- Total step
  - First, computation of the M-point DFTs

$$F(l,q)\equiv\sum_{m=0}^{M-1}x(l,m)W_M^{mq},\quad 0\le q\le M-1$$

  - Second, computation of a new array

$$G(l,q)=W_N^{lq}F(l,q),\quad 0\le l\le L-1,\ 0\le q\le M-1$$

  - Finally, computation of the L-point DFTs

$$X(p,q)=\sum_{l=0}^{L-1}G(l,q)W_L^{lp},\quad 0\le p\le L-1$$

8.1.2 Divide-and-Conquer Approach

- Computational complexity

  Complex multiplications: $N(M+L+1)$

  Complex additions: $N(M+L-2)$

  where $N = ML$

- For example, suppose that N=1000 and we select L=2 and M=500
  - Direct computation: 10^6
  - Divide-and-Conquer: 50300

8.1.2 Divide-and-Conquer Approach

- Ex. 8.1.1
  - To illustrate this computational procedure, let us consider the computation of an N=15 point DFT. Since N=5x3=15, we select L=5 and M=3. In other words, we store the 15-point sequence x(n) column-wise as follows:

$$\begin{aligned}
&\text{Row 1: } x(0,0)=x(0)\ \ x(0,1)=x(5)\ \ x(0,2)=x(10)\\
&\text{Row 2: } x(1,0)=x(1)\ \ x(1,1)=x(6)\ \ x(1,2)=x(11)\\
&\text{Row 3: } x(2,0)=x(2)\ \ x(2,1)=x(7)\ \ x(2,2)=x(12)\\
&\text{Row 4: } x(3,0)=x(3)\ \ x(3,1)=x(8)\ \ x(3,2)=x(13)\\
&\text{Row 5: } x(4,0)=x(4)\ \ x(4,1)=x(9)\ \ x(4,2)=x(14)
\end{aligned}$$

8.1.2 Divide-and-Conquer Approach

- Ex. 8.1.1
  - Compute the 3-point DFTs ➔ multiply each of the term $F(l,q)$

$$\begin{matrix} F(0,0) & F(0,1) & F(0,2) \\ F(1,0) & F(1,1) & F(1,2) \\ F(2,0) & F(2,1) & F(2,2) \\ F(3,0) & F(3,1) & F(3,2) \\ F(4,0) & F(4,1) & F(4,2) \end{matrix}$$

Col 1	Col 2	Col 3
$G(0,0)$	$G(0,1)$	$G(0,2)$
$G(1,0)$	$G(1,1)$	$G(1,2)$
$G(2,0)$	$G(2,1)$	$G(2,2)$
$G(3,0)$	$G(3,1)$	$G(3,2)$
$G(4,0)$	$G(4,1)$	$G(4,2)$

8.1.2 Divide-and-Conquer Approach

- Ex. 8.1.1
  - The final step is to compute the 5-point DFTs for each of the 3 columns

$$\begin{matrix} X(0,0)=X(0) & X(0,1)=X(1) & X(0,2)=X(2) \\ X(1,0)=X(3) & X(1,1)=X(4) & X(1,2)=X(5) \\ X(2,0)=X(6) & X(2,1)=X(7) & X(2,2)=X(8) \\ X(3,0)=X(9) & X(3,1)=X(10) & X(3,2)=X(11) \\ X(4,0)=X(12) & X(4,1)=X(13) & X(4,2)=X(14) \end{matrix}$$

8.1.2 Divide-and-Conquer Approach

- Ex. 8.1.1
  - The final step is to compute the 5-point DFTs for each of the 3 columns

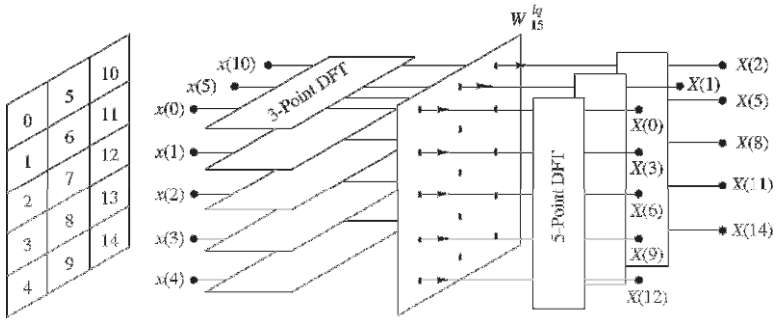


Figure 8.1.3 Computation of $N = 15$-point DFT by means of 3-point and 5-point DFTs.

8.1.2 Divide-and-Conquer Approach

- Algorithm 1.
  - Store the signal column-wise
  - Compute the M-point DFT of each row
  - Multiply the resulting array by the phase factors W_N^{lq}
  - Compute the L-point DFT of each column
  - Read the resulting array row-wise

Radix-2 FFT Algorithm

- Radix-2 FFT algorithm is
  - $N = r_1 r_2 r_3 ... r_v$
  - $r_1 = r_2 = r_3 = ... = r_v$ ← r is called the radix
- Derivation of a radix-2 algorithm
  - Decimation-in-time algorithm
    - M=N/2, L=2 (N=LM)
    - Now the N-point DFT can be expressed as

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

$$= \sum_{n \text{ even}} x(n) W_N^{kn} + \sum_{n \text{ odd}} x(n) W_N^{kn} = \sum_{m=0}^{N/2-1} x(2m) W_N^{2mk} + \sum_{m=0}^{N/2-1} x(2m+1) W_N^{k(2m+1)}$$

Radix-2 FFT Algorithm

- But $W_N^2 = W_{N/2}$, with this substitution, it can be expressed as

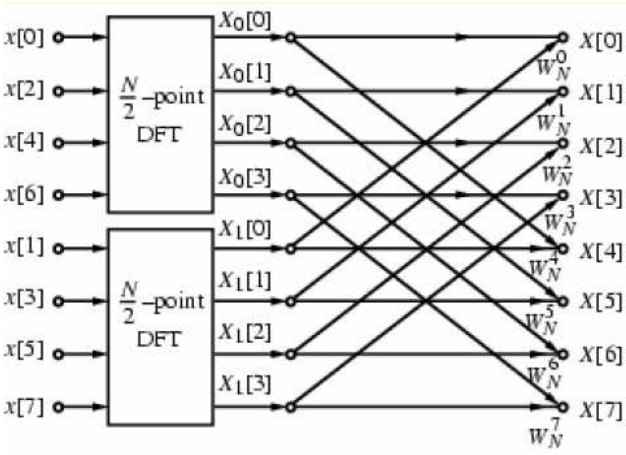
$$X(k) = \sum_{m=0}^{N/2-1} f_1(m) W_{N/2}^{km} + W_N^k \sum_{m=0}^{N/2-1} f_2(m) W_{N/2}^{km}$$

$$= F_1(k) + W_N^k F_2(k), \quad k = 0,1,...,N-1$$

where $f_1(n) = x(2n)$ and $f_2(n) = x(2n+1), \quad n = 0,1,...N/2-1$

- Note that $F_1(k)$ and $F_2(k)$ are the N/2-point DFTs

Radix-2 FFT Algorithm



Radix-2 FFT Algorithm

- In addition, the factor $W_N^{k+N/2} = -W_N^k$

$$X(k) = F_1(k) + W_N^k F_2(k), \quad k = 0,1,\ldots, N/2-1$$

$$X(k+\frac{N}{2}) = F_1(k) - W_N^k F_2(k), \quad k = 0,1,\ldots, N/2-1$$

$X(k)$ requires $2(N/2)^2 + N/2 = N^2/2 + N/2$ complex multiplications.

Radix-2 FFT Algorithm

- To be consistent with our previous notation, we may define

$$G_1(k) = F_1(k), \quad k = 0,1,\ldots, N/2-1$$
$$G_2(k) = W_N^k F_2(k), \quad k = 0,1,\ldots, N/2-1$$

- Then the DFT $X(k)$ may be expressed as

$$X(k) = G_1(k) + G_2(k), \quad k = 0,1,\ldots, N/2-1$$
$$X(k+\frac{N}{2}) = G_1(k) - G_2(k), \quad k = 0,1,\ldots, N/2-1$$

Radix-2 FFT Algorithm

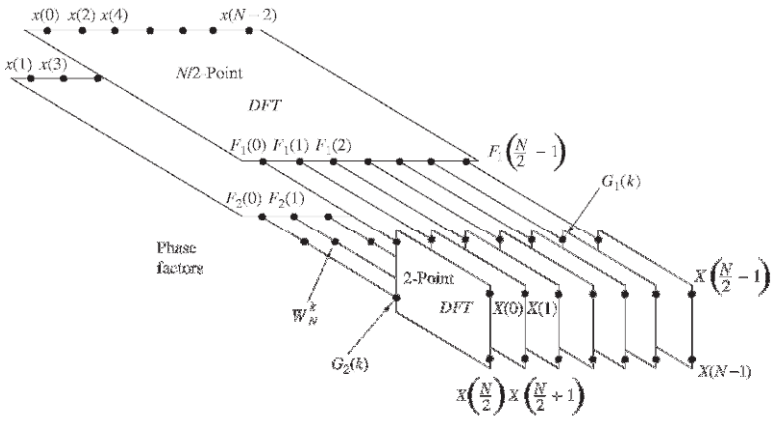


Figure 8.1.4 First step in the decimation-in-time algorithm.

Radix-2 FFT Algorithm

- Having performed the decimation-in-time once, we can repeat the process for each of the sequences $f_1(n)$ and $f_2(n)$

$$u_{11}(n) = f_1(2n), \qquad n = 0,1,...N/4-1$$
$$u_{12}(n) = f_1(2n+1), \qquad n = 0,1,...N/4-1$$

and $f_2(n)$ would yield

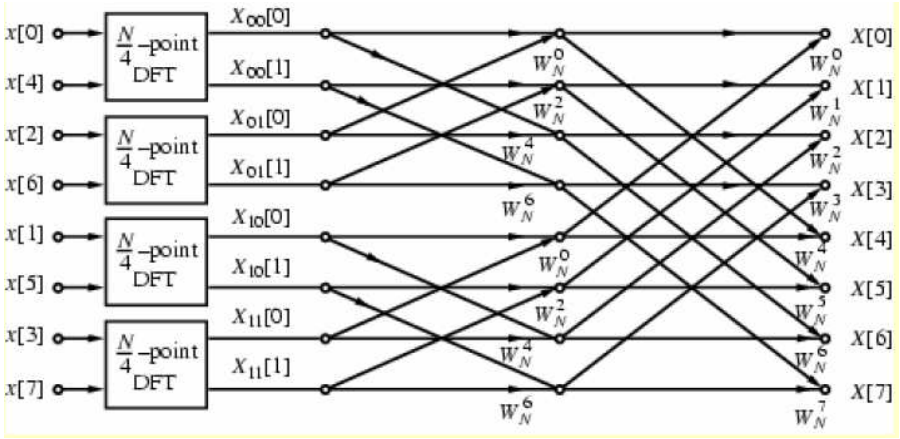
$$u_{21}(n) = f_2(2n), \qquad n = 0,1,...N/4-1$$
$$u_{22}(n) = f_2(2n+1), \qquad n = 0,1,...N/4-1$$

Radix-2 FFT Algorithm

- By computing N/4-point DFTs, we would obtain the N/2-point DFTs $F_1(k)$ and $F_2(k)$ from the relation

$$F_1(k) = V_{11}(k) + W_{N/2}^k V_{12}(k), \qquad k = 0,1,...,\frac{N}{4}-1$$
$$F_1(k+\frac{N}{4}) = V_{11}(k) - W_{N/2}^k V_{12}(k), \quad k = 0,1,...,\frac{N}{4}-1$$
$$F_2(k) = V_{21}(k) + W_{N/2}^k V_{22}(k), \qquad k = 0,1,...,\frac{N}{4}-1$$
$$F_2(k+\frac{N}{4}) = V_{21}(k) - W_{N/2}^k V_{22}(k), \quad k = 0,1,...,\frac{N}{4}-1$$

Radix-2 FFT Algorithm



8.1.3 Radix-2 FFT Algorithm

TABLE 8.1 Comparison of Computational Complexity for the Direct Computation of the DFT Versus the FFT Algorithm

Number of Points, N	Complex Multiplications in Direct Computation, N^2	Complex Multiplications in FFT Algorithm, $(N/2)\log_2 N$	Speed Improvement Factor
4	16	4	4.0
8	64	12	5.3
16	256	32	8.0
32	1,024	80	12.8
64	4,096	192	21.3
128	16,384	448	36.6
256	65,536	1,024	64.0
512	262,144	2,304	113.8
1,024	1,048,576	5,120	204.8

Radix-2 FFT Algorithm

- N=8-point DFT.
  - Four two-point DFT, two four-point DFT and finally, one eight-point DFT

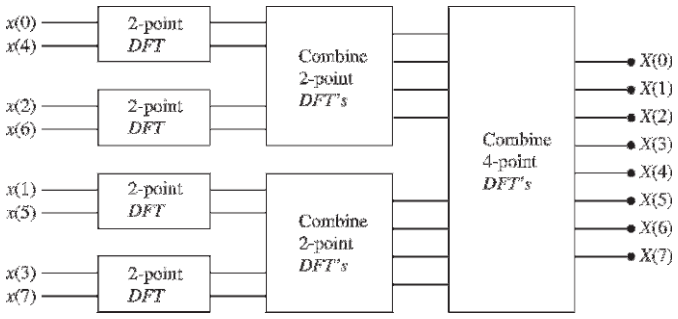


Figure 8.1.5 Three stages in the computation of an $N = 8$-point DFT.

Radix-2 FFT Algorithm

- The combination of the smaller DFTs to form the larger DFT

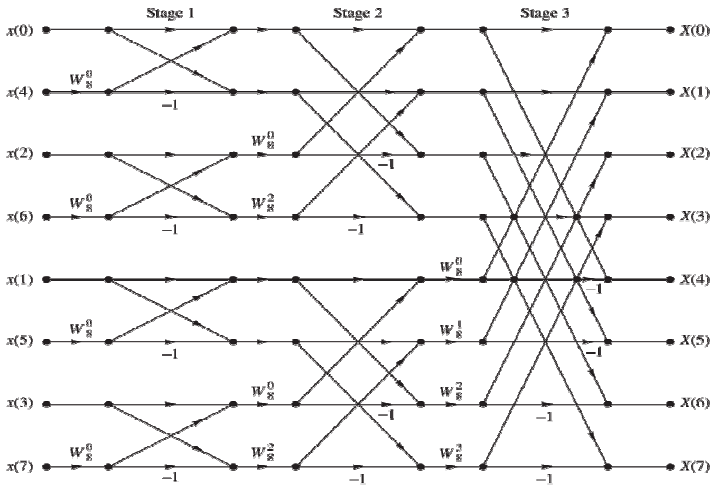


Figure 8.1.6 Eight-point decimation-in-time FFT algorithm.

Radix-2 FFT Algorithm

- The basic computation, which is shown in Fig. 8.1.7, is called a "*butterfly*"

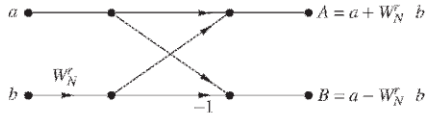


Figure 8.1.7 Basic butterfly computation in the decimation-in-time FFT algorithm.

- In general, each butterfly involves one complex multiplication and two complex and $\log_2 N$ stages.
  - Total numbers of complex multiplications and additions
    - $(N/2)\log_2 N$ and $N\log_2 N$

Radix-2 FFT Algorithm

Complex multiplications : $(N/2) \cdot \log_2 N$
Complex additions : $N \cdot \log_2 N$

TABLE 8.1 Comparison of Computational Complexity for the Direct Computation of the DFT Versus the FFT Algorithm

Number of Points, N	Complex Multiplications in Direct Computation, N^2	Complex Multiplications in FFT Algorithm, $(N/2)\log_2 N$	Speed Improvement Factor
4	16	4	4.0
8	64	12	5.3
16	256	32	8.0
32	1,024	80	12.8
64	4,096	192	21.3
128	16,384	448	36.6
256	65,536	1,024	64.0
512	262,144	2,304	113.8
1,024	1,048,576	5,120	204.8

8.1.3 Radix-2 FFT Algorithm

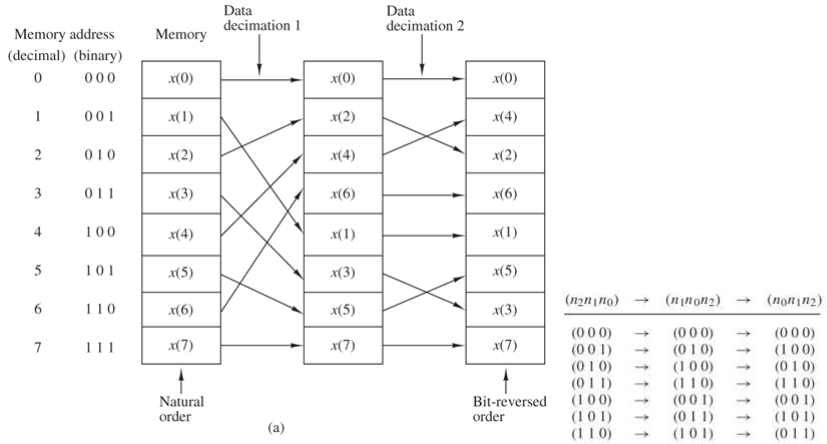


Figure 8.1.8 Shuffling of the data and bit reversal.

Radix-2 FFT Algorithm

- Another important radix-2 FFT algorithm, called the decimation-in-frequency algorithm
  - Using divide-and-conquer approach
  - M=2 and L=N/2
- Thus, we obtain

$$X(k) = \sum_{n=0}^{N/2-1} x(n)W_N^{kn} + \sum_{n=N/2}^{N-1} x(n)W_N^{kn}$$

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

Radix-2 FFT Algorithm

- Since $W_N^{kN/2} = (-1)^k$, the expression can be rewritten as

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

Now, let us split $X(k)$ into the even- abd odd-numbered samples.

$$X(2k) = \sum_{n=0}^{N/2-1} \left[x(n) + x(n+\frac{N}{2}) \right] W_{N/2}^{kn}, \quad k = 0,1,...,\frac{N}{2}-1$$

and

$$X(2k+1) = \sum_{n=0}^{N/2-1} \left\{ \left[x(n) - x(n+\frac{N}{2}) \right] W_N^n \right\} W_{N/2}^{kn}, \quad k = 0,1,...,\frac{N}{2}-1$$

Radix-2 FFT Algorithm

- If we define the $N/2$-point sequences $g_1(n)$ and $g_2(n)$ as

$$g_1(n) = x(n) + x(n+\frac{N}{2})$$

$$g_2(n) = \left[x(n) - x(n+\frac{N}{2}) \right] W_N^n, \quad n = 0,1,2,...\frac{N}{2}-1$$

then

$$X(2k) = \sum_{n=0}^{N/2-1} g_1(n) W_{N/2}^{kn}$$

$$X(2k+1) = \sum_{n=0}^{N/2-1} g_2(n) W_{N/2}^{kn}$$

Radix-2 FFT Algorithm

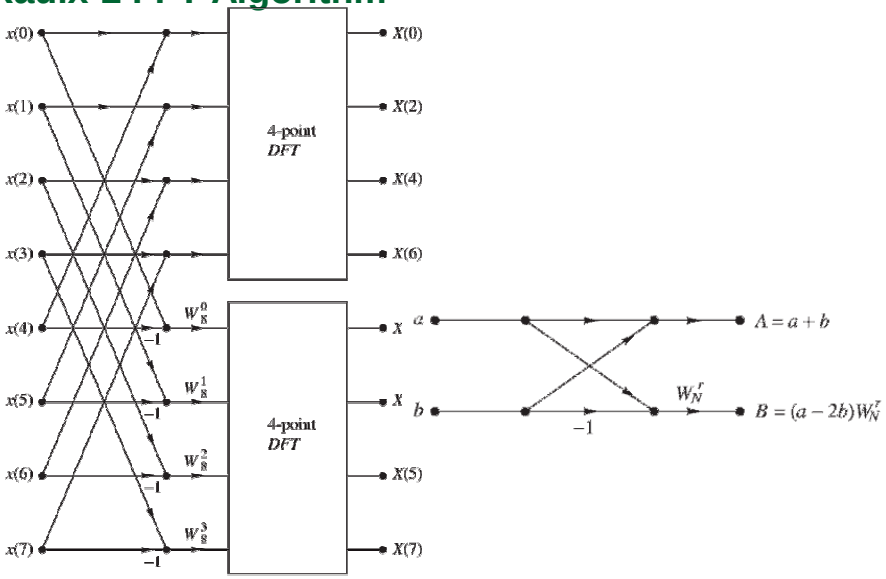


Figure 8.1.9 First stage of the decimation-in-frequency FFT algorithm.

8.1.3 Radix-2 FFT Algorithm

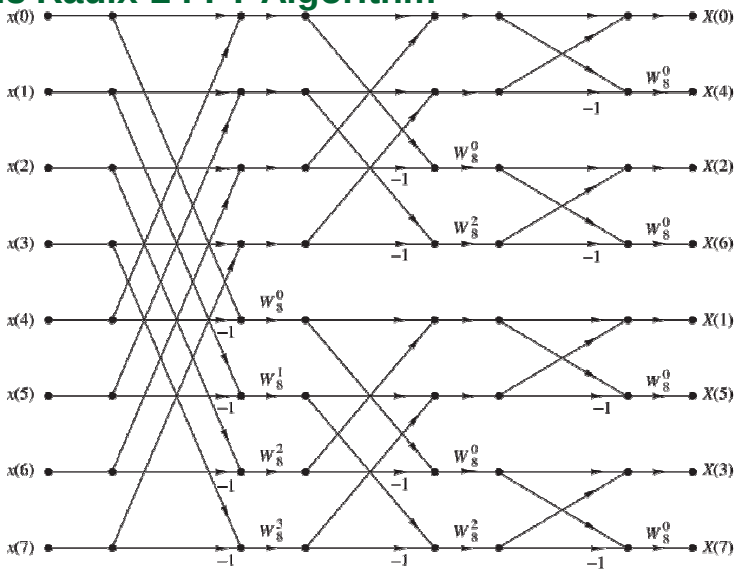


Figure 8.1.11 $N = 8$-point decimation-in-frequency FFT algorithm.

8.1.4 Radix-4 FFT Algorithm

- If the number of data points N in the DFT is a power of 4
  - Still can use radix-2 algorithm
  - Radix-4 algorithm is more efficient

$$X(p,q)=\sum_{l=0}^{3}\left[W_N^{lq}F(l,q)\right]W_4^{lp},\quad p=0,1,2,3$$

where $F(l,q)$ is given by (8.1.16)

$$F(l,q)=\sum_{m=0}^{N/4-1}x(l,m)W_{N/4}^{mq},\quad \begin{cases} l=0,1,2,3 \\ q=0,1,2,\ldots,\dfrac{N}{4}-1 \end{cases}$$

8.1.4 Radix-4 FFT Algorithm

- and

$$x(l,m)=x(4m+l)$$

$$X(p,q)=X\left(\frac{N}{4}p+q\right)$$

- Thus, the four $N/4$-point DFTs obtained from (8.1.40) are combined according to (8.1.39) to yield the N-point DFT
  - Radix-4 decimation-in-time butterfly

$$\begin{bmatrix} X(0,q) \\ X(1,q) \\ X(2,q) \\ X(3,q) \end{bmatrix}=\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}\begin{bmatrix} W_N^{0}F(0,q) \\ W_N^{q}F(1,q) \\ W_N^{2q}F(2,q) \\ W_N^{3q}F(3,q) \end{bmatrix}$$

8.1.4 Radix-4 FFT Algorithm

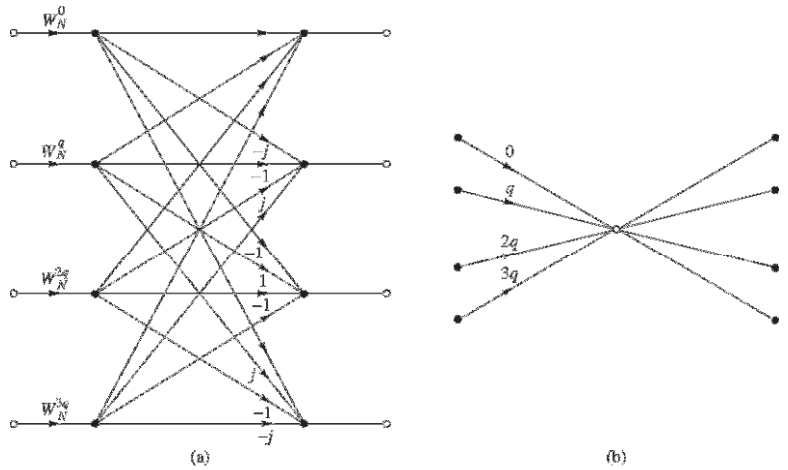


Figure 8.1.12 Basic butterfly computation in a radix-4 FFT algorithm.

8.1.4 Radix-4 FFT Algorithm

- It is possible to reduce the number of additions per butterfly from 12 to 8
  - By expressing the matrix of the linear transformation in (8.1.43) as a product
-

$$\begin{bmatrix} X(0,q) \\ X(1,q) \\ X(2,q) \\ X(3,q) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -j \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & j \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} W_N^0 F(0,q) \\ W_N^q F(1,q) \\ W_N^{2q} F(2,q) \\ W_N^{3q} F(3,q) \end{bmatrix}$$

8.1.4 Radix-4 FFT Algorithm

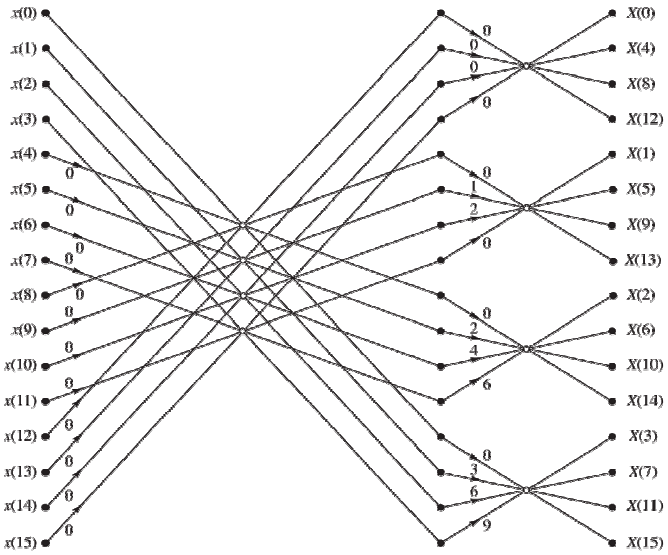


Figure 8.1.13 Sixteen-point radix-4 decimation-in-time algorithm with input in normal order and output in digit-reversed order. The integer multipliers shown on the graph represent the exponents on W_{16}.

8.1.4 Radix-4 FFT Algorithm

- An *radix-4 decimation-in-frequency* FFT can be obtained by selecting $L=N/4$. $M=4$, l, $p=0,1,\ldots,N/4-1$; m, $q=0,1,2,3$; $n=(N/4)m+l$; and $k=4p+q$.
- Choice of the parameters
  - The general equation can be expressed as

$$X(p,q)=\sum_{l=0}^{N/4-1} G(l,q)W_{N/4}^{lp}$$

$$\text{where } G(l,q)=W_N^{lq}F(l,q), \quad \begin{cases} q=0,1,2,3 \\ l=0,1,\ldots,N/4-1 \end{cases}$$

8.1.4 Radix-4 FFT Algorithm

- Choice of the parameters

$$F(l,q)=\sum_{m=0}^{3}x(l,m)W_4^{mq}, \qquad \begin{cases} q=0,1,2,3 \\ l=0,1,\ldots,N/4-1 \end{cases}$$

We note that $X(p,q)=X(4p+q),\ q=0,1,2,3.$

- Consequently, the N-point DFT is decimated into four $N/4$-point DFTs.

8.1.4 Radix-4 FFT Algorithm

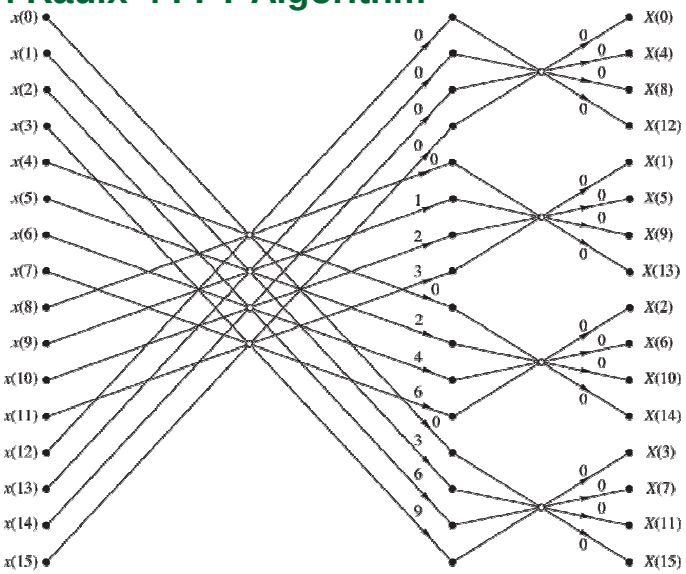


Figure 8.1.14 Sixteen-point, radix-4 decimation-in-frequency algorithm with input in normal order and output in digit-reversed order.

8.1.4 Radix-4 FFT Algorithm

- Re-derive the radix-4 decimation-in-frequency algorithm by breaking the N-point DFT into four smaller DFTs

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

$$=\sum_{n=0}^{N/4-1}x(n)W_N^{kn}+\sum_{n=N/4}^{N/2-1}x(n)W_N^{kn}+\sum_{n=N/2}^{3N/4-1}x(n)W_N^{kn}+\sum_{n=3N/4}^{N-1}x(n)W_N^{kn}$$

$$=\sum_{n=0}^{N/4-1}x(n)W_N^{kn}+W_N^{Nk/4}\sum_{n=0}^{N/4-1}x(n+\frac{N}{4})W_N^{kn}$$

$$+W_N^{kN/2}\sum_{n=0}^{N/4-1}x(n+\frac{N}{2})W_N^{kn}+W_N^{3kN/4}\sum_{n=0}^{N/4-1}x(n+\frac{3N}{4})W_N^{kn}$$

8.1.4 Radix-4 FFT Algorithm

- From the definition of the phase factors, we have

$$W_N^{kN/4}=(-j)^k,\quad W_N^{Nk/2}=(-1)^k,\quad W_N^{3Nk/4}=(j)^k$$

After substitution of (8.1.49) into (8.1.48), we obtain

$$X(k)=\sum_{n=0}^{N/4-1}[x(n)+(-j)^k x(n+\frac{N}{4})+$$

$$(-1)^k x(n+\frac{N}{2})+(j)^k x(n+\frac{3N}{4})]W_N^{nk}$$

- The relation in (8.1.50) is not an $N/4$-point DFT because the phase factor depends on N and on $N/4$

8.1.4 Radix-4 FFT Algorithm

- To convert it into an $N/4$-point DFT
  - Subdivide DFT sequence into $X(4k)$, $X(4k+1)$, $X(4k+2)$, and $X(4k+3)$, $k=0,1,\ldots,N/4-1$.

$$X(4k)=\sum_{n=0}^{N/4-1}[x(n)+x(n+\frac{N}{4})+x(n+\frac{N}{2})+x(n+\frac{3N}{4})]W_N^{0}W_{N/4}^{kn}$$

$$X(4k+1)=\sum_{n=0}^{N/4-1}[x(n)-jx(n+\frac{N}{4})-x(n+\frac{N}{2})+jx(n+\frac{3N}{4})]W_N^{n}W_{N/4}^{kn}$$

$$X(4k+2)=\sum_{n=0}^{N/4-1}[x(n)-x(n+\frac{N}{4})+x(n+\frac{N}{2})-x(n+\frac{3N}{4})]W_N^{2n}W_{N/4}^{kn}$$

$$X(4k+3)=\sum_{n=0}^{N/4-1}[x(n)+jx(n+\frac{N}{4})-x(n+\frac{N}{2})-jx(n+\frac{3N}{4})]W_N^{3n}W_{N/4}^{kn}$$

8.1.5 Split-Radix FFT Algorithms

- Basic idea
  - Use different computational methods for *independent parts* of the algorithm with the objective of reducing the # of computations
  - Recall that in the radix-2 decimation-in-frequency FFT algorithm
    - Even-numbered samples of the N-point DFT are given as

$$X(2k)=\sum_{n=0}^{N/2-1}[x(n)+x(n+\frac{N}{2})]W_{N/2}^{nk},\quad k=0,1,\ldots,\frac{N}{2}-1$$

    - Note that thee DFT points can be obtained from an $N/2$-point DFT without any additional multiplications.

8.1.5 Split-Radix FFT Algorithms

- If we use a radix-4 decimation-in-frequency FFT algorithm for the odd-numbered samples of the N-point DFT

$$X(4k+1)=\sum_{n=0}^{N/4-1}\left\{[x(n)-x(n+\frac{N}{2})]-j[x(n+\frac{N}{4})-x(n+\frac{3N}{4})]\right\}W_N^n W_{N/4}^{kn}$$

$$X(4k+3)=\sum_{n=0}^{N/4-1}\left\{[x(n)-x(n+\frac{N}{2})]+j[x(n+\frac{N}{4})-x(n+\frac{3N}{4})]\right\}W_N^{3n} W_{N/4}^{kn}$$

- N-point DFT is decomposed into one $N/2$-point DFT without additional phase factors and two $N/4$-point DFTs with phase factors

8.1.5 Split-Radix FFT Algorithms

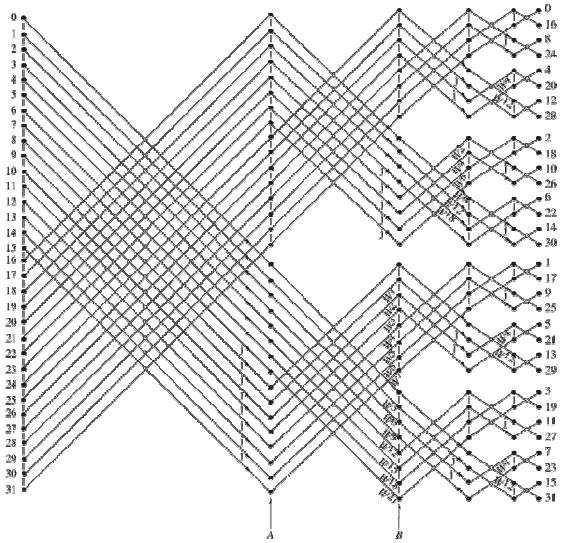


Figure 8.1.15 Length 32 split-radix FFT algorithms from paper by Duhamel (1986); reprinted with permission from the IEEE.

8.1.5 Split-Radix FFT Algorithms

- At stage A of the computation for N=32, the top 16 points constitute the sequence

$$g_0(n) = x(n) + x(n+N/2), \quad 0 \le n \le 15$$

- The next 8 points constitute the sequence

$$g_1(n) = x(n) - x(n+N/2), \quad 0 \le n \le 7$$

- The bottom eight points constitute the sequence $jg_2(n)$, where

$$g_2(n) = x(n+N/4) - x\left(n+3N/4\right), \quad 0 \le n \le 7$$

8.1.5 Split-Radix FFT Algorithms

- For a 16-point DFT at stage A
  - We decompose the computation into an eight-point, radix-2 DFT and two four-point radix-4 DFTs.
- At stage B
  - Top eight points constitute the sequence

$$g_0'(n) = g_0(n) + g_0(n+N/2), \quad 0 \le n \le 7$$

  - and the next eight points constitute the two four-point sequences $g_1'(n)$ and $jg_2'(n)$

$$g_1'(n) = g_0(n) - g_0(n+N/2), \quad 0 \le n \le 3$$
$$g_2'(n) = g_0(n+N/4) - g_0(n+3N/4), \quad 0 \le n \le 3$$

8.1.5 Split-Radix FFT Algorithms

- The bottom 16 points of stage B are in the form of two eight-point DFTs.
  - Decomposed into
    - Four-point, radix-2 DFT and four-point, radix-4 DFT

TABLE 8.2 Number of Nontrivial Real Multiplications and Additions to Compute an N-point Complex DFT

	Real Multiplications				Real Additions			
N	Radix 2	Radix 4	Radix 8	Split Radix	Radix 2	Radix 4	Radix 8	Split Radix
16	24	20		20	152	148		148
32	88			68	408			388
64	264	208	204	196	1,032	976	972	964
128	712			516	2,504			2,308
256	1,800	1,392		1,284	5,896	5,488		5,380
512	4,360		3,204	3,076	13,566		12,420	12,292
1,024	10,248	7,856		7,172	30,728	28,336		27,652

Source: Extracted from Duhamel (1986).

8.1.6 Implementation of FFT Algorithms

- Radix-2 FFT algorithm
  - Perform butterfly computations of two data points from memory
    - Repeated many times
  - Efficient implementation
    - Phase factors $\{W_N^k\}$ are computed first and stored
  - If the number of data points is not a power of 2
    - It is a simple matter to pad the sequence with zeros such that $N=2^v$

8.2 Applications of FFT Algorithms

- 8.2.1. efficient computation of the DFT of two real sequences
  - Input data may be real valued.
    - After phase factors at first stage, all variables are basically complex valued.
- Suppose that $x_1(n)$ and $x_2(n)$ are two real-valued sequences of length N, and let $x(n)$ be a complex-valued sequence defined as

$$x(n) = x_1(n) + jx_2(n), \quad 0 \le n \le N-1$$

- The DFT operation is linear and hence the DFT of $x(n)$ can be expressed as

$$X(k) = X_1(k) + jX_2(k)$$

8.2.1. Efficient Computation of The DFT of Two Real Sequences

- The sequences $x_1(n)$ and $x_2(n)$ can be expressed as

$$x_1(n) = \frac{x(n) + x^*(n)}{2}$$

$$x_2(n) = \frac{x(n) - x^*(n)}{2j}$$

  - Hence the DFTs of $x_1(n)$ and $x_2(n)$ are

$$X_1(k) = \frac{1}{2}\{\mathrm{DFT}[x(n)] + \mathrm{DFT}[x^*(n)]\}$$

$$X_2(k) = \frac{1}{2j}\{\mathrm{DFT}[x(n)] - \mathrm{DFT}[x^*(n)]\}$$

8.2.1. Efficient Computation of The DFT of Two Real Sequences

- Recall that the DFT of $x^*(n)$ is $X^*(N-k)$. Therefore,

$$X_1(k) = \frac{1}{2}[X(k) + X^*(N-k)]$$

$$X_2(k) = \frac{1}{2j}[X(k) - X^*(N-k)]$$

- By performing a single DFT on the complex-valued sequence $x(n)$
  - We have obtained the DFT of the two real sequences with only a small amount of additional computation.

8.2.2. Efficient Computation of The DFT of 2N-point Real Sequence

- Suppose that $g(n)$ is a real-valued sequence of $2N$ points

$$x_1(n) = g(2n)$$

$$x_2(n) = g(2n+1)$$

- Thus we have subdivided the $2N$-point real sequence into two N-point ones.
  - Let $x(n)$ be the N-point complex-valued sequence

$$x(n) = x_1(n) + jx_2(n)$$

$$X_1(k) = \frac{1}{2}[X(k) + X^*(N-k)]$$

$$X_2(k) = \frac{1}{2j}[X(k) - X^*(N-k)]$$

8.2.2. Efficient Computation of The DFT of 2*N*-point Real Sequence

- We must express the 2*N*-point DFT in terms of the two *N*-point DFTs.

$$G(k)=\sum_{n=0}^{N-1}g(2n)W_{2N}^{2nk}+\sum_{n=0}^{N-1}g(2n+1)W_{2N}^{(2n+1)k}$$

$$=\sum_{n=0}^{N-1}x_1(n)W_N^{nk}+W_{2N}^{k}\sum_{n=0}^{N-1}x_2(n)W_N^{nk}$$

Consequently,

$$G(k)=X_1(k)+W_2^k NX_2(k),\qquad k=0,1,\ldots,N-1$$

$$G(k+N)=X_1(k)-W_2^k NX_2(k),\qquad k=0,1,\ldots,N-1$$

8.2.3. Use of the DDT Algorithm in Linear Filtering and Correlation

- An important application of the FFT is in FIR linear filtering of long data sequences

Let $h(n),\ 0\le n\le M-1$, be the unit sample response of the FIR filter

$x(n)$ denote the input data sequence

The block size of the FFT algorithm is $N=L+M-1$

L : number of new data samples being processed by the filter

- Assume:
  - Given any value of M, and L is selected so that N is a power of 2

8.2.3. Use of the DDT Algorithm in Linear Filtering and Correlation

- Overlap-save method:
  - First M-1 data points of each data block are the last M-1 data points of the previous data block
  - Each block contains L new data points
    - $N=L+M-1$
    - Perform FFT on each block
- Overlap-add method
  - The computational method using the FFT is basically the same
  - Difference:
    - N-point data blocks consist of L new data points and M-1 additional zeros

8.2.3. Use of the DDT Algorithm in Linear Filtering and Correlation

- Computational complexity issue
  - Suppose that $M=128=2^7$ and $N=2^v$.
  - The number of complex multiplications per output point for an FFT size of $N=2^v$ is

$$c(v)=\frac{N\log_2 2N}{L}=\frac{2^v(v+1)}{N-M+1}$$
$$\approx\frac{2^v(v+1)}{2^v-2^7}$$

8.2.3. Use of the DDT Algorithm in Linear Filtering and Correlation

- Computational complexity issue

TABLE 8.3 Computational Complexity

Size of FFT $\nu = \log_2 N$	$c(\nu)$ Number of Complex Multiplications per Output Point
9	13.3
10	12.6
11	12.8
12	13.4
14	15.1

8.3 A linear Filter Approach to Computation of the DFT

- Some applications only require a selected number of values of the DFT
  - If the number is less tan $\log_2 N$, then a direct computation of the desired values is more efficient
- 8.3.1 The Goertzel algorithm
  - The Goertzel algorithm exploits the periodicity of the phase factor
    - Can be expressed as a linear filtering operation
  - Since $W_N^{-kN} = 1$, thus

$$X(k) = W_N^{-kN} \sum_{m=0}^{N-1} x(m) W_N^{km} = \sum_{m=0}^{N-1} x(m) W_N^{-k(N-m)}$$

8.3.1 The Goertzel algorithm

- If we define the sequence $y_k(n)$ as

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

- Then it is clear that $y_k(n)$ is the convolution of the finite-duration input sequence $x(n)$ of length N with a filter

$$h_k(n) = W_N^{-kn} u(n)$$

- The output of this filter at n=N yields the value of the DFT at $\omega_k = 2\pi k / N$

$$X(k) = y_k(n)|_{n=N}$$

8.3.1 The Goertzel algorithm

- The filter with impulse response $h_k(n)$ has the system function

$$H_k(z) = \frac{1}{1 - W_N^{-k} z^{-1}}$$

  - A pole on the unit circle at $\omega_k = 2\pi k / N$
  - Block ➔ parallel bank of N single-pole filters
- We can use difference equation to compute $y_k(n)$ recursively

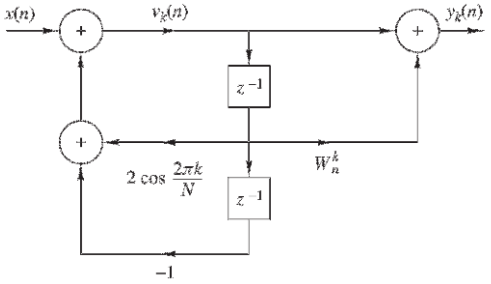
$$y_k(n) = W_N^{-k} y_k(n-1) + x(n), \quad y_k(-1) = 0$$

  - The desired output is $X(k) = y_k(N), \quad \text{for } k = 0,1,\ldots,N-1$

8.3.1 The Goertzel algorithm

- This leads to two-pole filters with system function of the form

$$H_k(z) = \frac{1 - W_N^k z^{-1}}{1 - 2\cos(2\pi k / N) z^{-1} + z^{-2}}$$



$$u_k(n) = 2\cos\frac{2\pi k}{N} v_k(n-1)$$

$$-v_k(n-2) + x(n)$$

$$y_k(n) = v_k(n) - W_N^k v_k(n-1)$$

with initial condifions

$$v_k(-1) = v_k(-2) = 0$$

Figure 8.3.1 Direct form II realization of two-pole resonator for computing the DFT.

8.3.2 The chirp-z transform Algorithm

- Suppose that we wish to compute the values of the z-transform of $x(n)$ at a set of points $\{z_k\}$

$$X(z_k) = \sum_{n=0}^{N-1} x(n) z_k^{-n}, \qquad k = 0,1,\ldots,L-1$$

- If the contour is a circle of radius r and the z_k are N equally spaces points, then

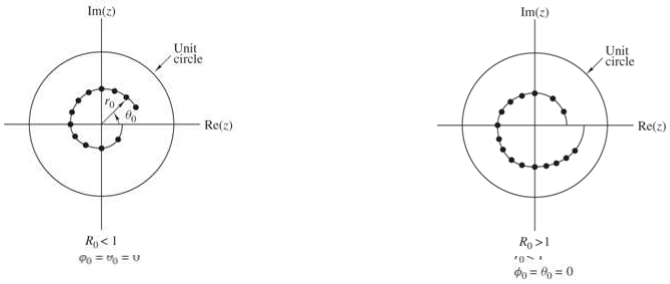
$$z_k = re^{j2\pi kn/N}, \qquad k = 0,1,2,\ldots,N-1$$

$$X(z_k) = \sum_{n=0}^{N-1} [x(n) r^{-n}] e^{-j2\pi kn/N}, \qquad k = 0,1,2,\ldots,N-1$$

8.3.2 The chirp-z transform Algorithm

- More generally, suppose that the points z_k in the z-plane fall on an arc which begins at some point

$$z_0 = r_0 e^{j\theta_0}$$
$$z_k = r_0 e^{j\theta_0}(R_0 e^{j\phi_0})^k, \qquad k = 0,1,\ldots,L-1$$



8.3.2 The chirp-z transform Algorithm

- When the points $\{z_k\}$ in (8.3.12) are substituted into the expression for the z-transform, we obtain

$$X(z_k) = \sum_{n=0}^{N-1} x(n) z_k^{-n}$$
$$= \sum_{n=0}^{N-1} x(n)(r_0 e^{j\theta_0})^{-n} V^{-nk}$$

  - Where, by definition, $V = R_0 e^{j\phi_0}$
  - We can express (8.3.13) in the form of a convolution

$$nk = \frac{1}{2}[n^2 + k^2 - (k-n)^2]$$
$$X(z_k) = V^{-k^2/2} \sum_{n=0}^{N-1} [x(n)(r_0 e^{j\theta_0})^{-n} V^{-n^2/2}] V^{(k-n)^2/2}$$

8.3.2 The chirp-z transform Algorithm

- Let us define a new sequence $g(n)$ as

$$g(n) = x(n)(r_0 e^{j\theta_0})^{-n} V^{-n^2/2}$$

Then (8.3.16) can be expressed as

$$X(z_k) = V^{-k^2/2} \sum_{n=0}^{N-1} g(n) V^{(k-n)^2/2}$$

- It can be interpreted as the convolution of $g(n)$ and impulse response $h(n) = V^{n^2/2}$
-

$$X(z_k) = V^{-k^2/2} y(k) = \frac{y(k)}{h(k)}, \quad k = 0,1,\ldots,L\text{-}1$$

$$y(k) = \sum_{n=0}^{N-1} g(n) h(k-n), \quad k = 0,1,\ldots,L-1$$

8.3.2 The chirp-z transform Algorithm

- The linear convolution in (8.3.21) is most efficiently done by use of the FFT algorithm
  - Let us consider the circular convolution of N-point sequence with an M-point section of $h(n)$
  - We should select a DFT of size: $M=L+N-1$, which would yield L valid points and N-1 points corrupted by aliasing

$$h_1(n) = h(n-N+1), \quad n = 0,1,\ldots,M-1$$

and compute its M-point DFT via the FFT algorithm to obtain $H_1(k)$

  - From $x(n)$ compute $g(n)$ as specified by (8.3.17), pad $g(n)$ with L-1 zeros
    - Compute its M-point DFT to yield $G(k)$

8.3.2 The chirp-z transform Algorithm

- The first N-1 points of $y_1(n)$ are corrupted by aliasing and are discarded.
  - The desired values are $y_1(n)$ for $N-1 \le n \le M-1$
  - Correspond to the range $0 \le n \le L-1$
  - $y(n) = y_1(n+N-1), \quad n = 0,1,...,L-1$
- Alternatively, we can define a sequence $h_2(n)$ as

$$h_2(n) = \begin{cases} h(n), & 0 \le n \le L\text{-}1 \\ h(n-N-L+1), & L \le n \le M\text{-}1 \end{cases}$$

- The M-point DFT of $h_2(n)$ yields $H_2(K)$

$$Y_2(k) = G(k)H_2(k)$$

8.3.2 The chirp-z transform Algorithm

- The IDFT of $Y_2(k)$ yields the sequence $y_2(n)$ for $0 \le n \le M-1$
  - Now the desired values of $y_2(n)$ are in the range $0 \le n \le L-1$

$$y(n) = y_2(n), \quad n = 0,1,...,L-1$$

  - Finally, the complex values $X(z_k)$ are computed by dividing $y(k)$ by $h(k)$, k=0,1,…,L-1.
- For the computation of DFT, we select

$$r_0 = R_0 = 1, \ \theta_0 = 0, \ \phi_0 = 2\pi / N, \text{ and } L = N$$

$$V^{-n^2/2} = e^{-j\pi n^2/N} = \cos\frac{\pi n^2}{N} - j\sin\frac{\pi n^2}{N}$$

  - The chirp filter with impulse response

$$h(n) = V^{n^2/2} = \cos\frac{\pi n^2}{N} - j\sin\frac{\pi n^2}{N} = h_r(n) + jh_i(n)$$

8.3.2 The chirp-z transform Algorithm

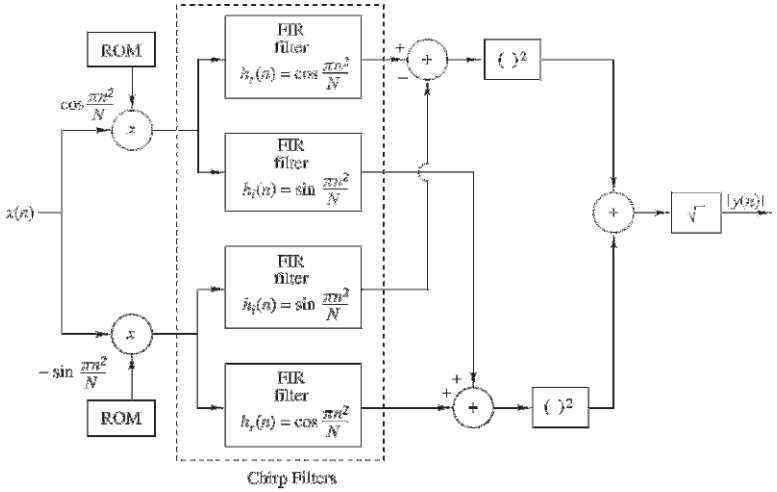


Figure 8.3.3 Block diagram illustrating the implementation of the chirp-z transform for computing the DFT (magnitude only).

8.4.1 Quantization Errors in the Direct Computation of the DFT

- Given a finite-duration sequence $\{x(n)\}$, $0 \le n \le N-1$
  - The DFT of $\{x(n)\}$ is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad k = 0,1,\ldots,N-1, \quad \text{where } W_N = e^{-j2\pi/N}$$

  - The quantization errors:
    - The quantization errors due to rounding are uniformly distributed random variables in the range $(-\Delta/2, \Delta/2)$ where $\Delta = 2^{-b}$
    - The $4N$ quantization errors are mutually uncorrelated
    - The $4N$ quantization errors are uncorrelated with the sequence $\{x(n)\}$

8.4.1 Quantization Errors in the Direct Computation of the DFT

- Variance of quantization errors

$$\sigma_e^2 = \frac{\Delta^2}{12} = \frac{2^{-2b}}{12}$$

and the variance of the quantization errors from the $4N$ multiplications is

$$\sigma_q^2 = 4N\sigma_e^2 = \frac{N}{3} \cdot 2^{-2b}$$

- When N is a power of 2, variance can be expressed as

$$\sigma_q^2 = \frac{2^{-2(b-v/2)}}{3}$$

8.4.1 Quantization Errors in the Direct Computation of the DFT

- Clearly, an upper bound on $|X(k)|$ is

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

If the dynamic range in addition is (-1,1), then $|X(k)| < 1$ requires that

$$\sum_{n=0}^{N-1} |x(n)| < 1$$

- If $|x(n)|$ is initially scaled such that $|x(n)|<1$ for all n
- Each point in the sequence can be divided by N to ensure that (8.4.6) is satisfied.

8.4.1 Quantization Errors in the Direct Computation of the DFT

- The scaling implied by (8.4.6) is extremely severe.
  - If the $\{x(n)\}$ is white and then, after scaling, each value $|(x(n)|$ of the sequence is uniformly distributed in the range (-1/N,1/N)

$$\sigma_x^2 = \frac{(2/N)^2}{12} = \frac{1}{3N^2}$$

  - And the variance of the output DFT coefficients is

$$\sigma_X^2 = N\sigma_x^2 = \frac{1}{3N}$$

  - The signal-to-noise power ratio is

$$\frac{\sigma_X^2}{\sigma_q^2} = \frac{2^{2b}}{N^2}$$

8.4.1 Quantization Errors in the Direct Computation of the DFT

- Scaling is responsible for reducing the SNR by N
- Scaling + quantization errors result in a total reduction by N^2
- Ex. 8.4.1
  - Use (8.4.9) to determine the number of bits required to compute the DFT of a 1024-point sequence with a SNR of 30dB
  - Sol: The size of the sequence is $N=2^{10}$. Hence the SNR is

$$10\log_{10}\frac{\sigma_X^2}{\sigma_q^2} = 10\log_{10} 2^{2b-20}$$

8.4.1 Quantization Errors in the Direct Computation of the DFT

- Ex. 8.4.1
  - Sol. (cont.) : For an SNR of 30 dB, we have

$$3(2b-20)=30, \quad b=15 \text{ bits}.$$

  - Note that the 15 bits is the precision for both multiplication and addition.
- Suppose we simply require that $|x(n)|<1$
  - Must provide a sufficiently large dynamic range for addition such that $|X(k)|<N$

8.4.1 Quantization Errors in the Direct Computation of the DFT

- In previous case, the variance of the sequence $\{|x(n)|\}$ is $1/3$
  - The variance of $|X(k)|$ is $\sigma_X^2 = N\sigma_x^2 = \frac{N}{3}$
  - Consequently, the SNR is $\frac{\sigma_X^2}{\sigma_q^2} = 2^{2b}$
- If we repeat the computation in Ex. 8.4.1
  - b=5 bits
  - However, we need an additional 10 bits for accumulator
  - Precision in multiplication from 15 bits to 5 bits.

8.4.2 Quantization Errors in the Direct Computation of the FFT

- The FFT algorithms require significantly fewer multiplications that the direct computation of the DFT.
  - Result in smaller quantization errors?
    - No
  - Let us consider the use of fixed-point arithmetic in the computation radix-2 FFT
    - N=8

8.4.2 Quantization Errors in the Direct Computation of the FFT

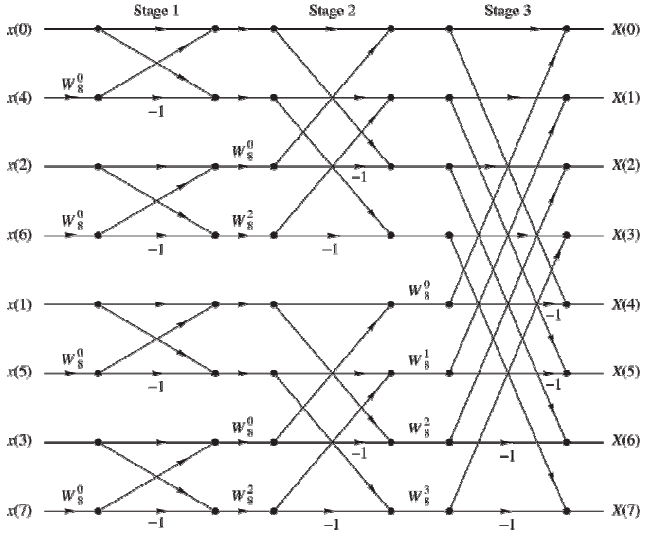


Figure 8.4.1 Decimation-in-time FFT algorithm.

8.4.2 Quantization Errors in the Direct Computation of the FFT

- Each butterfly computation involves one complex-valued multiplication or four real multiplications.
- There are $N/2$ in the first stage of the FFT
  - $N/4$ in the second stage
  - $N/8$ in the third stage, and so on.
- The number of butterflies per output point is

$$2^{v-1}+2^{v-2}+\ldots+2+1=2^{v-1}[1+(\frac{1}{2})+\ldots+(\frac{1}{2})^{v-1}]$$

$$=2^{v}[1-(\frac{1}{2})^{v}]=N-1$$

8.4.2 Quantization Errors in the Direct Computation of the FFT

- The butterflies that affect the computation of X(3) in the eight-point FFT of Fig. 8.4.1 are illustrated in Fig. 8.4.2

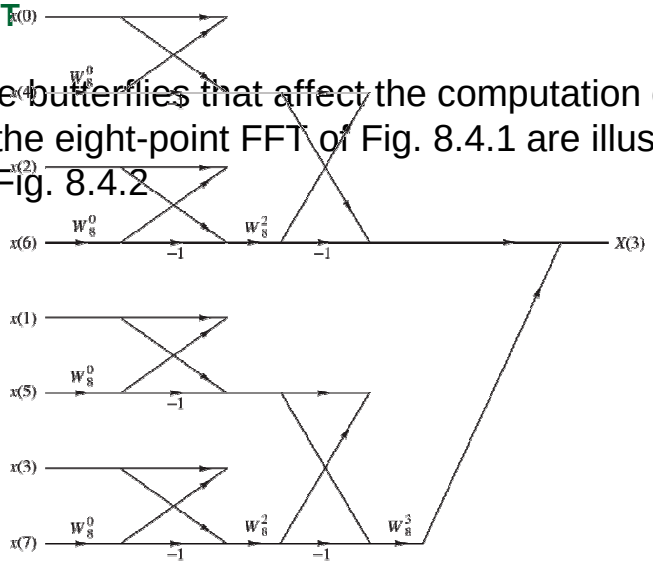


Figure 8.4.2 Butterflies that affect the computation of $X(3)$.

8.4.2 Quantization Errors in the Direct Computation of the FFT

- The quantization errors will be propagate to the output
  - They are phase shifted (phase rotated) by the phase factor W_N^{kn}
  - Assume that the quantization errors in each butterfly are uncorrelated with the errors in other butterflies
    - There are 4(N-1) errors that affect the output of each point of the FFT
  - The variance of total quantization error at output is

$$\sigma_q^2 = 4(N-1)\frac{\Delta^2}{12} \approx \frac{N\Delta^2}{3}, \quad \text{where } \Delta = 2^{-b}$$

8.4.2 Quantization Errors in the Direct Computation of the FFT

- Hence $\sigma_q^2 = \frac{N}{3}2^{-2b}$
- This is exactly the same result that we obtained for the direct computation of the DFT!
  - Due to FFT doesn't reduce the number of multiplications required to compute a single point of the DFT
- Scaling issue
  - The same SNR is obtained for the FFT
  - It is possible to devise a different scaling strategy that is not as severe as dividing each input point by N

8.4.2 Quantization Errors in the Direct Computation of the FFT

- Alternative scaling strategy

$$\max[|X_{n+1}(k)|,|X_{n+1}(l)|] \geq \max[|X_n(k)|,|X_n(l)|]$$
$$\max[|X_{n+1}(k)|,|X_{n+1}(l)|] \leq 2\max[|X_n(k)|,|X_n(l)|]$$

- We can distribute the total scaling of $1/N$ into each of the stages of the FFT
  - If $|x(n)|<1$ ➔ scale factor at first stage is 1/2
  - After v stages we have achieved an overall scale factor of $1/N$

8.4.2 Quantization Errors in the Direct Computation of the FFT

- Total variance of the quantization errors at the output of FFT is

$$\sigma_q^2 = \frac{\Delta^2}{12}\left\{4(\frac{N}{2})(\frac{1}{4})^{v-1} + 4(\frac{N}{4})(\frac{1}{4})^{v-2} + 4(\frac{N}{8})(\frac{1}{4})^{v-3} + \ldots + 4\right\}$$
$$= \frac{\Delta^2}{3}\left\{(\frac{1}{2})^{v-1} + (\frac{1}{2})^{v-2} + \ldots + \frac{1}{2} + 1\right\}$$
$$= \frac{2\Delta^2}{3}\left\{(1-(\frac{1}{2})^{v}\right\} \approx \frac{2}{3}2^{-2b}$$

- The signal has the variance $\sigma_X^2 = 1/3N$. SNR is

$$\frac{\sigma_X^2}{\sigma_q^2} = \frac{1}{2N}2^{2b} = 2^{2b-v-1}$$

8.4.2 Quantization Errors in the Direct Computation of the FFT

- Ex. 6.4.2
  - Determine the number of bits required to compute an FFT of 1024 points with an SNR of 30dB when the scaling is distributed as described above.
- Sol: The size of the FFT is $N=2^{10}$. Hence SNR according to (8.4.17) is

$$10\log_{10} 2^{2b-v-1} = 30$$

$$3(2b-11) = 30, \quad b = \frac{21}{2}(11 \text{ bits})$$

  - This can be compared with the 15 bits required if all the scaling is performed in the first stage of the FFT algorithm