数字信号处理

任课教师：许录平 余航

单位：空间科学与技术学院

Email：hyu@xidian.edu.cn

第四章 快速傅里叶变换(FFT)

4.1 引言

4.2 基2FFT算法

4.3 进一步减少运算量的措施

引言

运算量		复数乘法	复数加法
$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{kn}$	1个 $X(k)$	N	N-1
	N个 $X(k)$	N^2	N(N-1)
		实数乘法	实数加法
$(a+jb)(c+jd)$	1次复乘	4	2
$=(ac-bd)+j(bc+ad)$	1次复加		2
	1个 $X(k)$	4N	2N+2(N-1)
	N个 $X(k)$	$4N^2$	2N(2N-1)

4.1 引言

DFT是信号分析与处理中的一种重要变换。因直接计算DFT的计算量与变换区间长度N的平方成正比，当N较大时，计算量太大，所以在快速傅里叶变换(简称FFT)出现以前，直接用DFT算法进行谱分析和信号的实时处理是不切实际的。直到1965年发现了DFT的一种快速算法以后，情况才发生了根本的变化。

基2FFT、基4FFT、分裂基FFT、DHT等。

引言

当 N 较大时，计算量太大，所以在快速傅里叶变换**(Fast Fourier Transform, FFT)**出现以前， 直接用DFT算法进行谱分析和信号的实时处理是不切实际的

引言

■ 降低运算量的途径

$$\mathbf{X} = \mathbf{W}_N \mathbf{x}$$

$$\mathbf{W}_N = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^1 & W_N^2 & \cdots & W_N^{(N-1)} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & & & & \\ 1 & W_N^{(N-1)} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix}$$

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

引言

- 降低运算量的途径

$$\mathbf{X} = \mathbf{W}_N \mathbf{x}$$

$$\mathbf{W}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

引言

■ 降低运算量的途径

$$\mathbf{X} = \mathbf{W}_N \mathbf{x}$$

$$\mathbf{W}_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & W_3^1 & W_3^2 \\ 1 & W_3^2 & W_3^4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & W_3^1 & W_3^2 \\ 1 & W_3^2 & W_3^1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & \dfrac{-1-j\sqrt{3}}{2} & \dfrac{-1+j\sqrt{3}}{2} \\ 1 & \dfrac{-1+j\sqrt{3}}{2} & \dfrac{-1-j\sqrt{3}}{2} \end{bmatrix}$$

引言

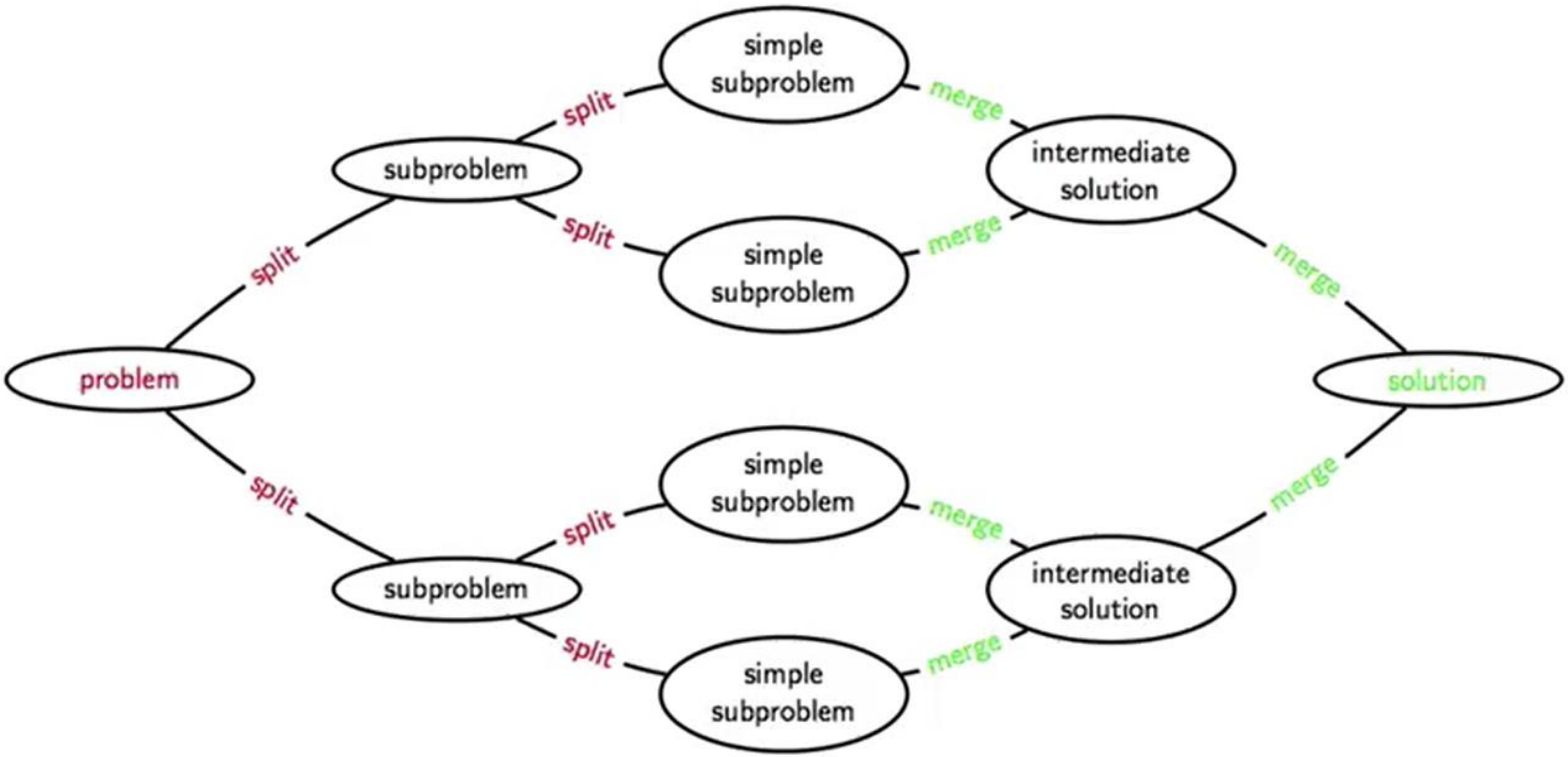
■ 降低运算量的途径

$$\mathbf{W}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_4^1 & W_4^2 & W_4^3 \\ 1 & W_4^2 & W_4^4 & W_4^6 \\ 1 & W_4^3 & W_4^6 & W_4^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_4^1 & W_4^2 & W_4^3 \\ 1 & W_4^2 & 1 & W_4^2 \\ 1 & W_4^3 & W_4^2 & W_4^1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

引言

- Divide and Conquer



引言

■ 降低运算量的途径

把N点DFT分解为几个较短的DFT， 可使乘法次数大大减少，另外， 利用旋转因子W_N^m的周期性、 对称性和可约性来减少DFT的运算次数

周期性：$W_N^{m+lN}=e^{-j\frac{2\pi}{N}(m+lN)}=e^{-j\frac{2\pi}{N}m}=W_N^m$

对称性：$W_N^{-m}=W_N^{N-m}$ $[W_N^{N-m}]^*=W_N^m$ $W_N^{m+\frac{N}{2}}=-W_N^m$

可约性：$W_N^m=W_{N/n}^{m/n}$

4.2 基2FFT算法

4.2.2 时域抽取法基2FFT基本原理

FFT算法基本上分为两大类：

- 时域抽取法FFT(Decimation In Time FFT,简称DIT-FFT)
- 频域抽取法FFT(Decimation In Frequency FFT,简称DIF—FFT)。

4.2 基2FFT算法

4.2.2 时域抽取法--基2FFT基本原理

设序列$x(n)$的长度为 N，且满足 $N=2^M, M$ 为正整数，如不满足，可补零，按n的奇偶把$x(n)$分解为两个$N/2$点的子序列

$$x_1(r)=x(2r), \qquad r=0,1,\cdots\frac{N}{2}-1$$

$$x_2(r)=x(2r+1), \quad r=0,1,\cdots\frac{N}{2}-1$$

4.2.2 时域抽取法基2FFT基本原理

$$X(k)=\sum_{n为偶}x(n)W_N^{kn}+\sum_{n为奇}x(n)W_N^{kn}$$

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

$$=\sum_{r=0}^{N/2-1}x_1(r)W_N^{2kr}+W_N^k\sum_{r=0}^{N/2-1}x_2(r)W_N^{2kr}$$

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

$$=X_1(k)+W_N^kX_2(k),\qquad k=0,1,\cdots,N-1$$

4.2.2 时域抽取法基2FFT基本原理

$$X(k)=\sum_{n\text{为偶}} x(n)W_N^{kn}+\sum_{n\text{为奇}} x(n)W_N^{kn}$$

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

$$=\sum_{r=0}^{N/2-1} x_1(r)W_N^{2kr}+W_N^k\sum_{r=0}^{N/2-1} x_2(r)W_N^{2kr}$$

$$=X_1(k)+W_N^k X_2(k),\qquad k=0,1,\cdots,N-1$$

其中 $$\boxed{W_N^{2kr}=e^{-j\frac{2\pi}{N}2kr}=e^{-j\frac{2\pi}{\frac{N}{2}}kr}=W_{N/2}^{kr}}$$ ⟵ **旋转因子可约性**

$$X_1(k)=\sum_{r=0}^{N/2-1} x_1(r)W_{N/2}^{kr}=DFT[x_1(r)]_{N/2}\qquad(4.2.5)$$

$$X_2(k)=\sum_{r=0}^{N/2-1} x_2(r)W_{N/2}^{kr}=DFT[x_2(r)]_{N/2}\qquad(4.2.6)$$

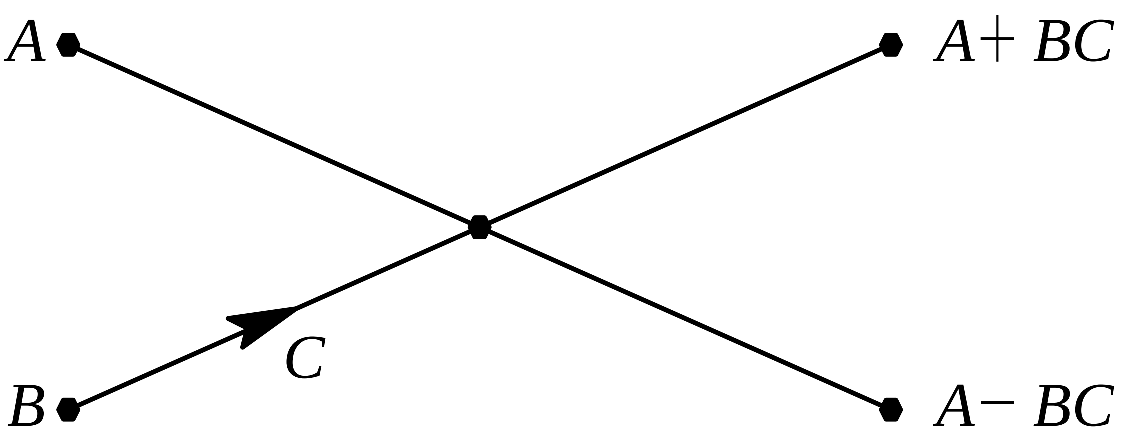
4.2.2 时域抽取法基2FFT基本原理

由于$X_1(k)$和$X_2(k)$均以$N/2$为周期，且 $W_N^{k+\frac{N}{2}} = -W_N^k$

所以$X(k)$又可表示为

$$X(k) = X_1(k) + W_N^k X_2(k) \quad k = 0,1,\cdots\frac{N}{2}-1 \tag{4.2.7}$$

$$X(k+\frac{N}{2}) = X_1(k) - W_N^k X_2(k) \quad k = 0,1,\cdots\frac{N}{2}-1 \tag{4.2.8}$$



蝶形单元需要
一次复数乘法
两次复数加法

图4.2.1 蝶形运算符号

4.2.2 时域抽取法基2FFT基本原理

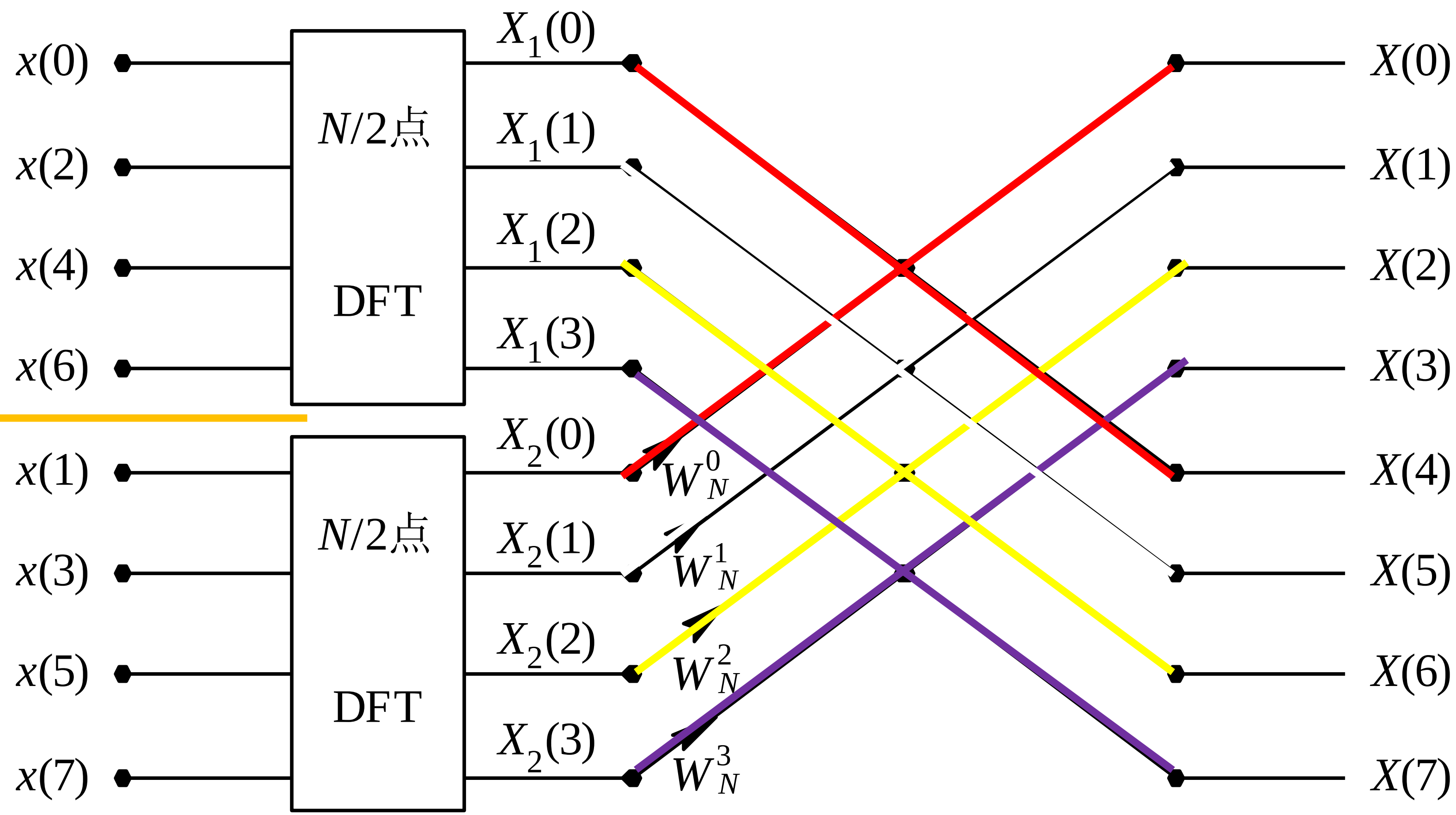


图4.2.2　N点DFT的一次时域抽取分解图(N=8)

一次分解
的运算量？

4.2.2 时域抽取法基2FFT基本原理

同理，将$x_1(r)$按奇偶分解成两个$N/4$长的子序列$x_3(l)$和$x_4(l)$，即

$$\left.\begin{aligned} x_3(l) &= x_1(2l) \\ x_4(l) &= x_1(2l+1) \end{aligned}\right\}, l = 0,1,\cdots,\frac{N}{4}-1$$

那么，$X_1(k)$又可表示为

$$\begin{aligned} X_1(k) &= \sum_{l=0}^{N/4-1} x_1(2l)W_{N/2}^{2kl} + \sum_{l=0}^{N/4-1} x_1(2l+1)W_{N/2}^{k(2l+1)} \\ &= \sum_{l=0}^{N/4-1} x_3(l)W_{N/4}^{kl} + W_{N/2}^{k} \sum_{l=0}^{N/4-1} x_4(l)W_{N/4}^{kl} \\ &= X_3(k) + W_{N/2}^{k} X_4(k), \quad k = 0,1,\cdots N/2-1 \end{aligned}$$

4.2.2 时域抽取法基2FFT基本原理

式中
$$X_3(k)=\sum_{l=0}^{N/4-1} x_3(l)W_{N/4}^{kl}=DFT[x_3(l)]$$
$$X_4(k)=\sum_{l=0}^{N/4-1} x_4(l)W_{N/4}^{kl}=DFT[x_4(l)] \tag{4.2.9}$$

$$\boxed{\left.\begin{aligned} X_1(k)&=X_3(k)+W_{N/2}^{k}X_4(k)\\ X_1(k+N/4)&=X_3(k)-W_{N/2}^{k}X_4(k)\end{aligned}\right\},k=0,1,\cdots,N/4-1} \tag{4.2.10}$$

4.2.2 时域抽取法基2FFT基本原理

同理可得：

$$\boxed{\left.\begin{aligned} X_2(k) &= X_5(k) + W_{N/2}^{k} X_6(k) \\ X_2(k+N/4) &= X_5(k) - W_{N/2}^{k} X_6(k) \end{aligned}\right\}, k = 0,1,\cdots N/4-1} \quad (4.2.11)$$

其中

$$X_5(k) = \sum_{l=0}^{N/4-1} x_5(l) W_{N/4}^{kl} = DFT[x_5(l)]$$

$$X_6(k) = \sum_{l=0}^{N/4-1} x_6(l) W_{N/4}^{kl} = DFT[x_6(l)]$$

$$\left.\begin{aligned} x_5(l) &= x_2(2l) \\ x_6(l) &= x_2(2l+1) \end{aligned}\right\}, l = 0,1,\cdots N/4-1$$

4.2.2 时域抽取法基2FFT基本原理

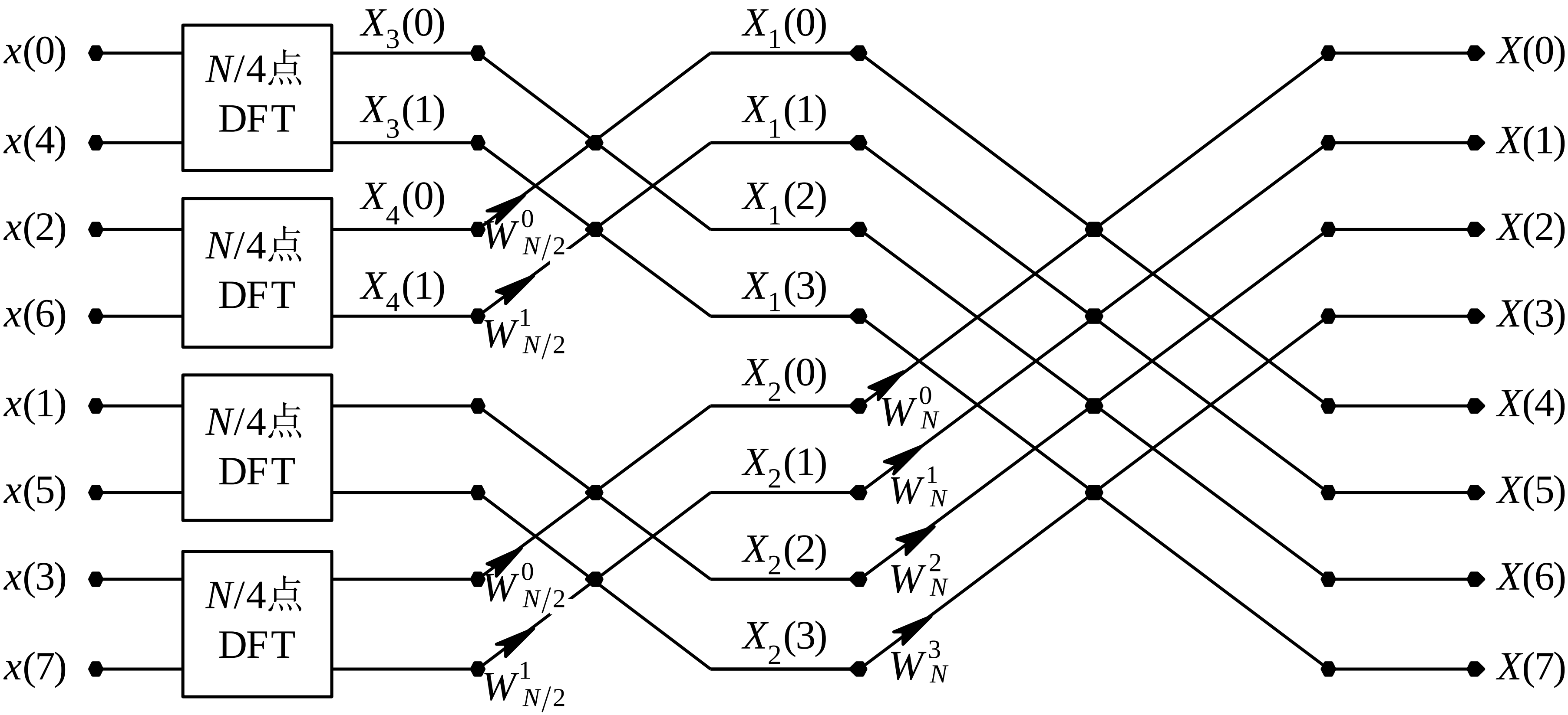
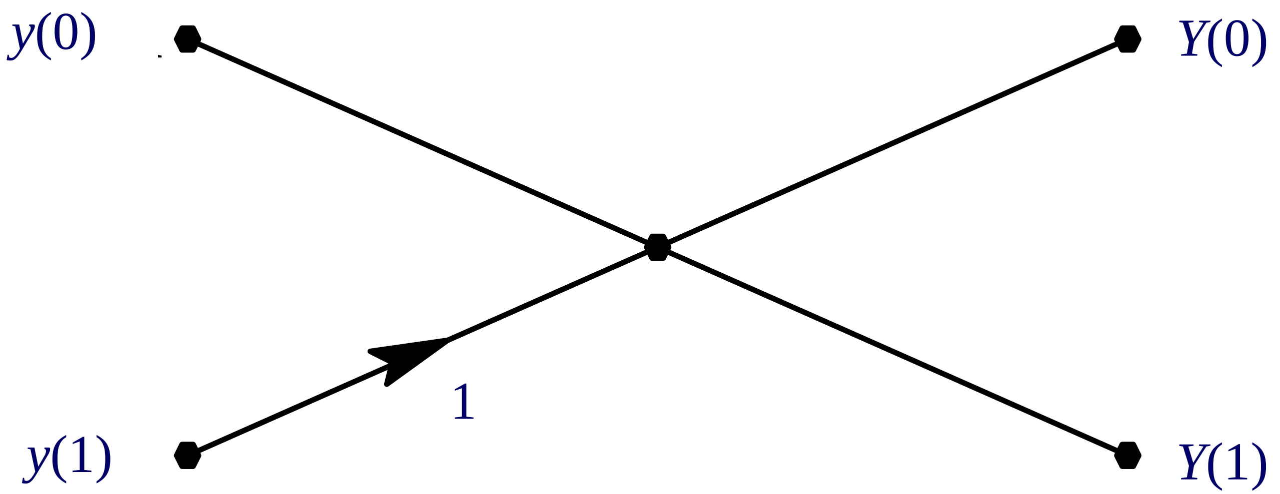


图4.2.3 8点DFT的第二次时域抽取分解图

4.2.2 时域抽取法基2FFT基本原理

- 2点DFT：

$$Y(k)=DFT[y(n)]=\sum_{n=0}^{1}y(n)W_2^{kn}=y(0)W_2^{k\cdot 0}+y(1)W_2^{k\cdot 1}$$



4.2.2 时域抽取法基2FFT基本原理

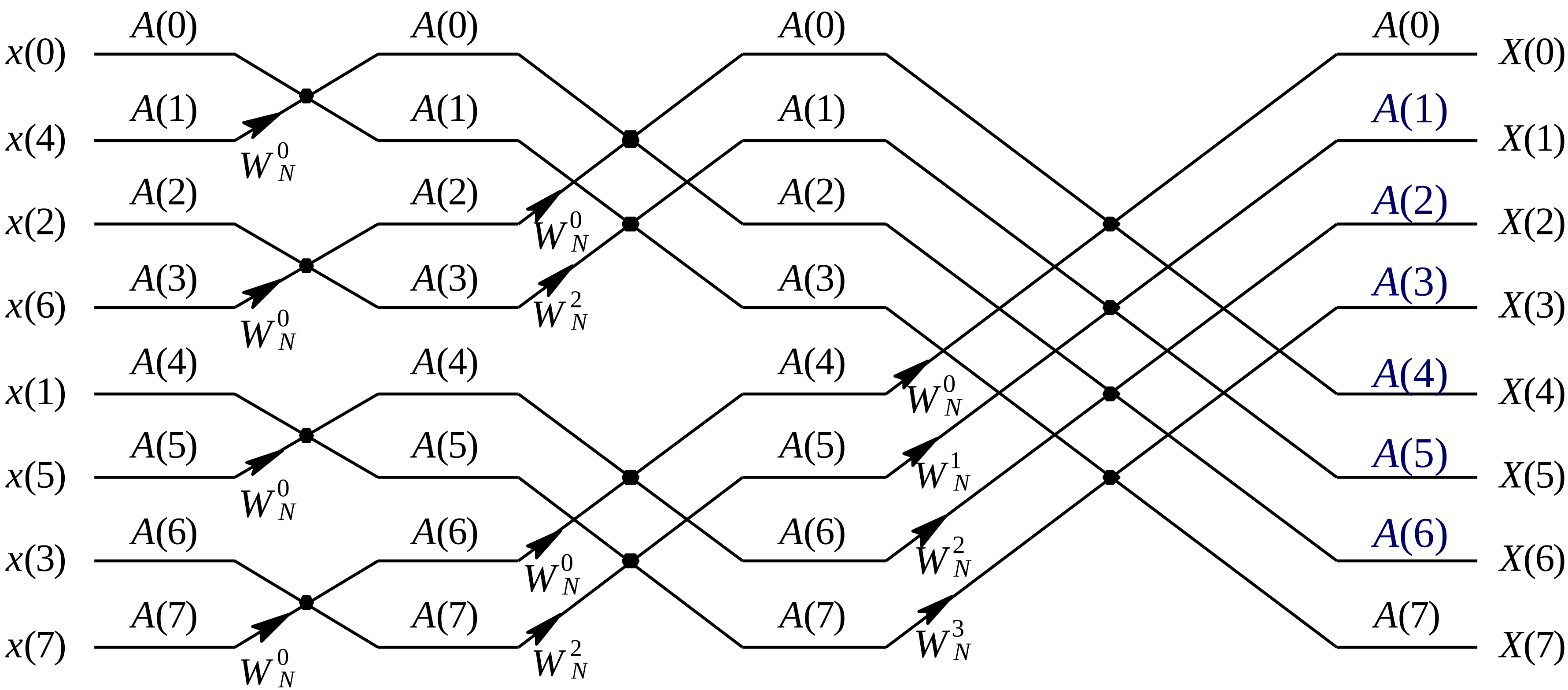


图4.2.4 N点DIT—FFT运算流图 (N=8)

A： 数组

4.2.3DIT-FFT算法与直接计算DFT运算量的比较

- 每一个蝶形：2次复数加法，1次复数乘法；
- 每一级运算需要N/2次蝶形运算；
- N点FFT共M级运算，$M=\log_2 N$。

所以，M级运算总共需要的复数乘次数为 $C_M = \frac{N}{2} \cdot M = \frac{N}{2}\log_2 N$

复数加次数为 $C_A = N \cdot M = N\log_2 N$

例如，$N=2^{10}=1024$时,复乘次数大大减少

$$\frac{N^2}{(N/2)\log_2 N} = \frac{1048576}{5120} = 204.8$$

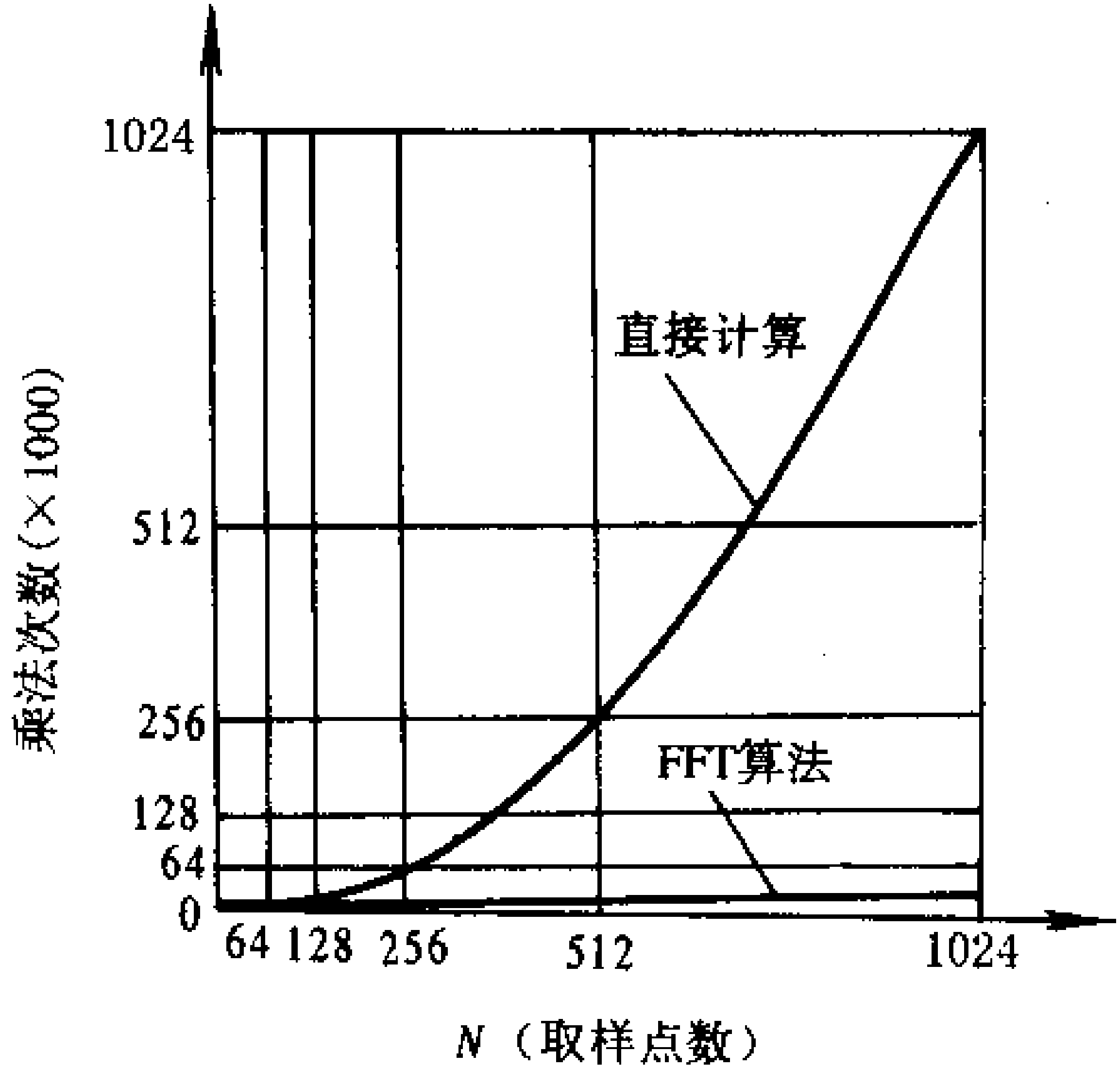
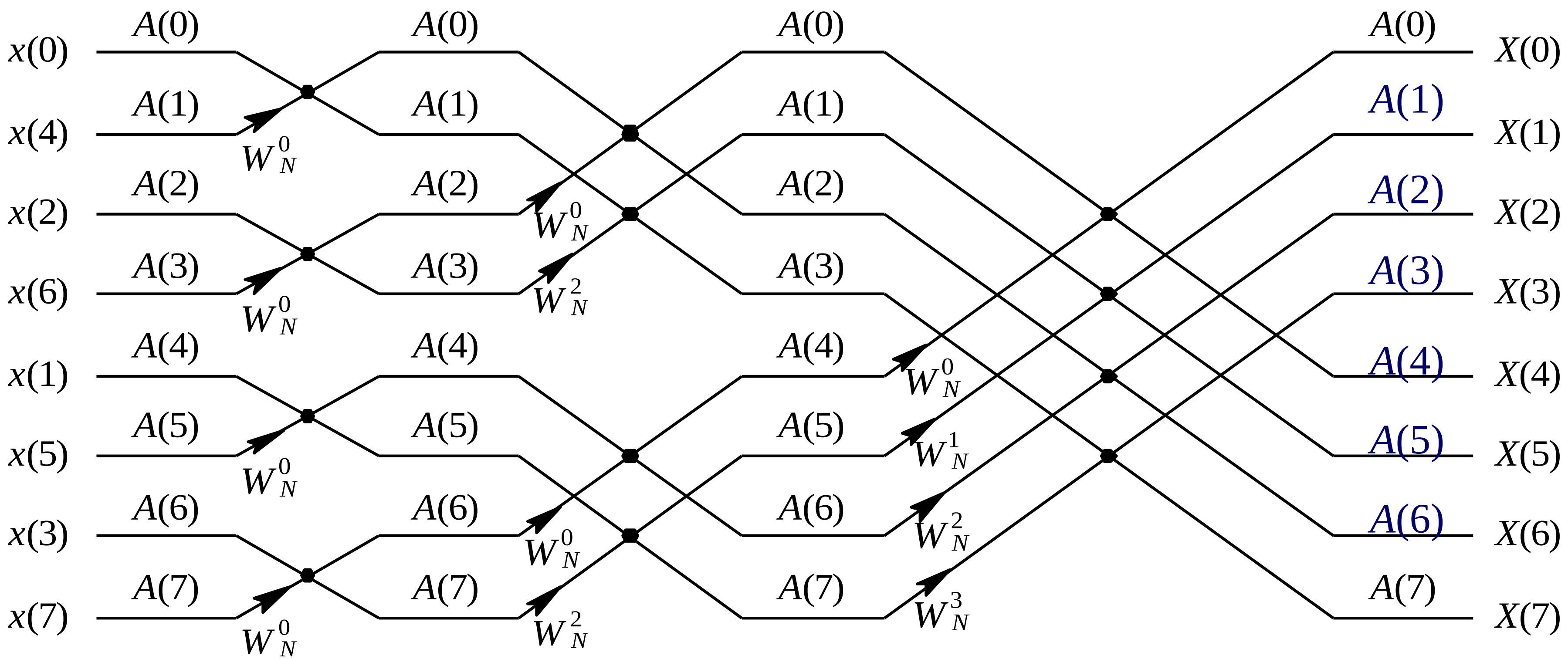


图4.2.5 FFT算法与直接计算DFT所需乘法次数的比较曲线

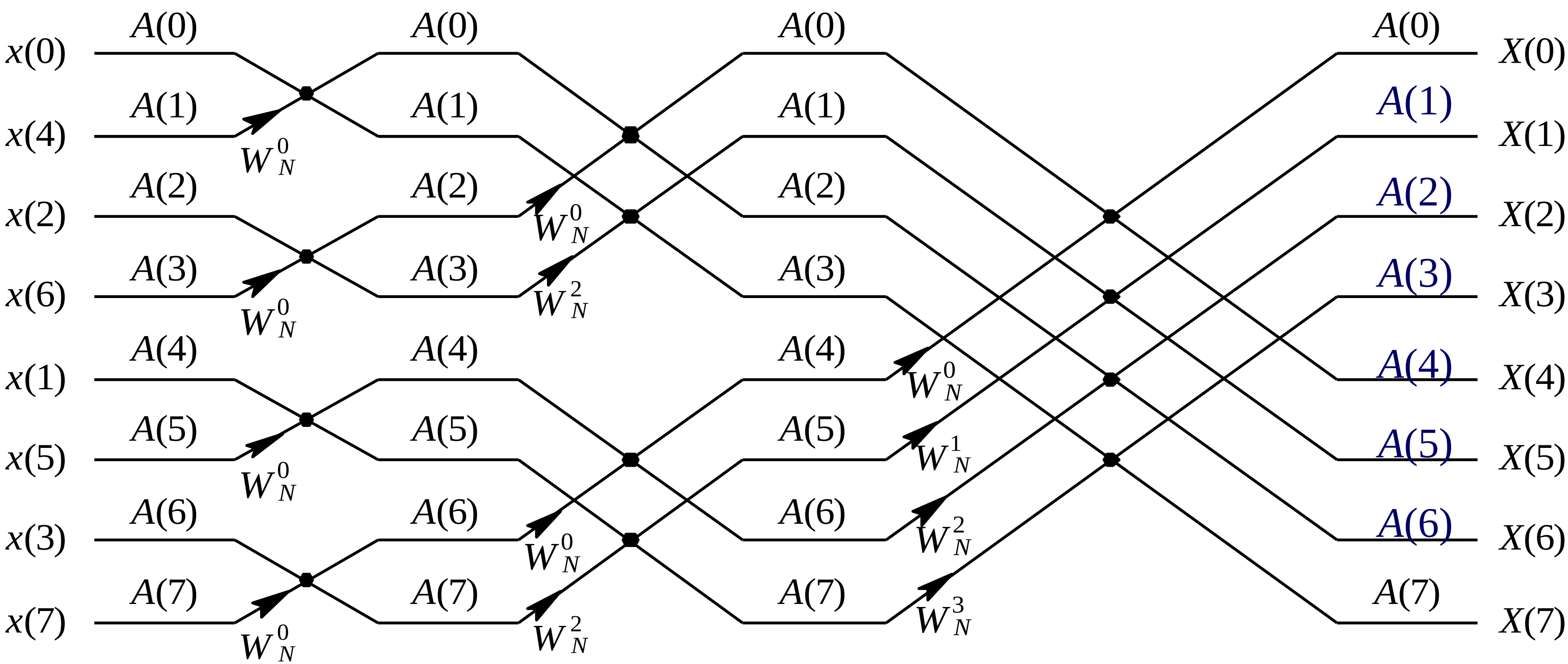
4.2.4 DIT-FFT的运算规律

1. 原位计算: 运算过程无需增加新的存储单元



4.2.4 DIT-FFT的运算规律及编程思想

2. 旋转因子的变化规律



- **N点DIT—FFT运算流图中，每级都有N/2个蝶形。**
- 每个蝶形都要乘以因子W_N^p ，称为旋转因子，*p*为旋转因子的指数。

4.2.4 DIT-FFT的运算规律

2. 旋转因子的变化规律

$N=2^3=8$时的各级旋转因子表示如下：

L=1时 $W_N^p = W_{N/4}^J = W_{2^L}^J, J=0$

L=2时 $W_N^p = W_{N/2}^J = W_{2^L}^J, J=0,1$

L=3时 $W_N^p = W_N^J = W_{2^L}^J, J=0,1,2,3$

对N=2^M的一般情况，第L级的旋转因子为

$$W_N^p = W_{2^L}^J \qquad J=0,1,2,\cdots,2^{L-1}-1$$

由于 $2^L = 2^M \times 2^{L-M} = N\cdot 2^{L-M}$

所以 $W_N^P = W_{N\cdot 2^{L-M}}^J = W_N^{J\cdot 2^{M-L}} \qquad J=0,1,2,\cdots,2^{L-1}-1$

$$p = J\cdot 2^{M-L}$$

3. 蝶形运算规律

设序列x(n)经时域抽选(倒序)后，存入数组X中。如果蝶形运算的两个输入数据相距B个点($B=2^{L-1}$)，应用原位计算，则蝶形运算可表示成如下形式：

$$X_L(J) \leftarrow X_{L-1}(J)+X_{L-1}(J+B)W^p_N$$

$$X_L(J+B) \leftarrow X_{L-1}(J)-X_{L-1}(J+B)W^p_N$$

式中下标L表示第L级运算，$X_L(J)$则表示第L级运算后数组元素X(J)的值。 $p=J\cdot 2^{M-L}$；$J=0,1,\ldots,2^{L-1}-1$；$L=1,2,\ldots,M$

4. 编程思想及程序框图

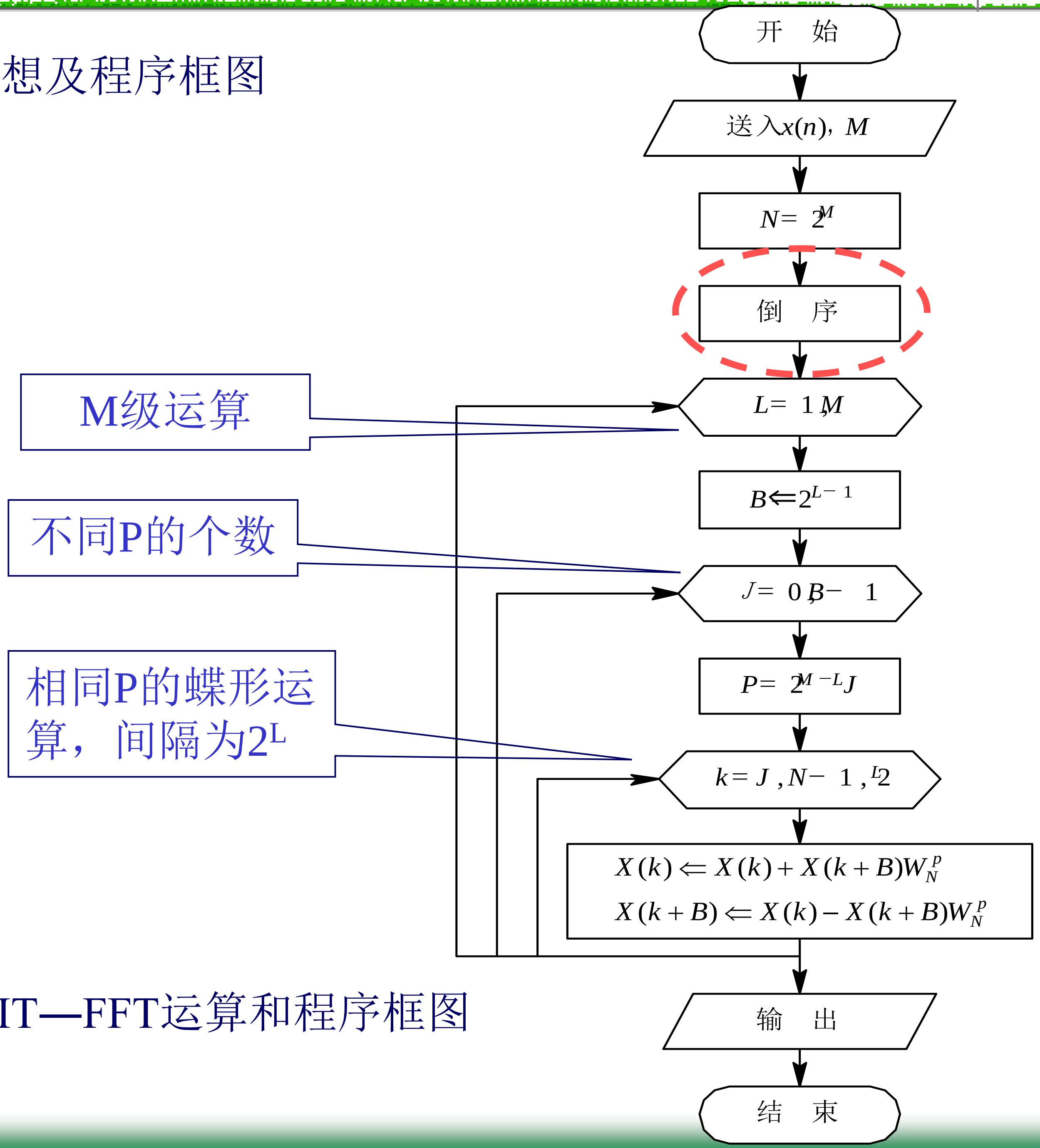
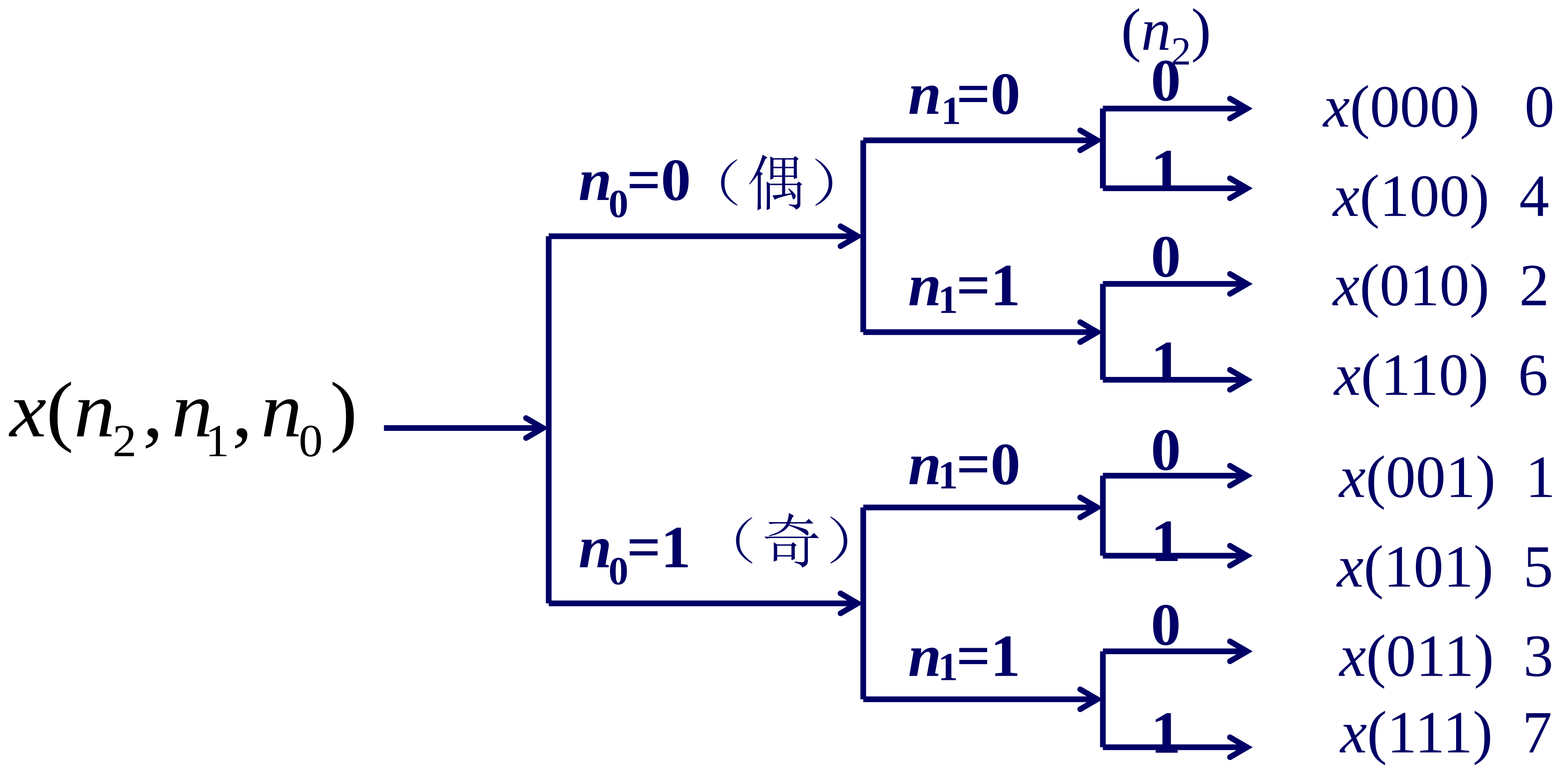


图4.2.6 DIT—FFT运算和程序框图

4.2.4 DIT−FFT的运算规律

5. 输入序列的倒序



4.2.4 DIT−FFT的运算规律

3. 输入序列的倒序

自然顺序n	二进制$n_2n_1n_0$	倒位序二进制$n_0n_1n_2$	倒位顺序$\hat{n}$
0	0 0 0	0 0 0	0
1	0 0 1	1 0 0	4
2	0 1 0	0 1 0	2
3	0 1 1	1 1 0	6
4	1 0 0	0 0 1	1
5	1 0 1	1 0 1	5
6	1 1 0	0 1 1	3
7	1 1 1	1 1 1	7

4.2.5 频域抽取法FFT(DIF–FFT)

设序列$x(n)$长度为$N=2^M$，首先将$x(n)$前后对半分开，得到两个子序列，其DFT可表示为：

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

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

4.2.5 频域抽取法FFT(DIF−FFT)

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

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

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

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

$$W_N^{kN/2}=(-1)^k=\begin{cases}1, & k=\text{偶数}\\ -1, & k=\text{奇数}\end{cases}$$

4.2.5 频域抽取法FFT(DIF–FFT)

将$X(k)$分解成偶数组与奇数组

- 当 k 取偶数($k = 2r,\ \ r = 0, 1, \ldots, N/2\text{-}1$)时

$$X(2r) = \sum_{n=0}^{N/2-1}[x(n) + x(n+\frac{N}{2})]W_N^{2nr}$$

$$= \sum_{n=0}^{N/2-1}[x(n) + x(n+\frac{N}{2})]W_{N/2}^{nr}$$

$$= \sum_{n=0}^{N/2-1} x_1(n) W_{N/2}^{nr}$$

其中 $x_1(n) = x(n) + x(n+\frac{N}{2}), \quad n = 0, 1, \cdots, N/2-1$

4.2.5 频域抽取法FFT(DIF–FFT)

将$X(k)$分解成偶数组与奇数组

- 当 k 取奇数($k = 2r+1$, $r = 0, 1,..., N/2-1$)时

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

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

$$=\sum_{n=0}^{N/2-1}[x(n)-x(n+\frac{N}{2})]W_N^n W_{N/2}^{nr}$$

$$=\sum_{n=0}^{N/2-1}x_2(n)W_{N/2}^{nr}$$

其中 $x_2(n)=[x(n)-x(n+\frac{N}{2})]W_N^n,\quad n=0,1,\cdots,N/2-1$

4.2.5 频域抽取法FFT(DIF−FFT)

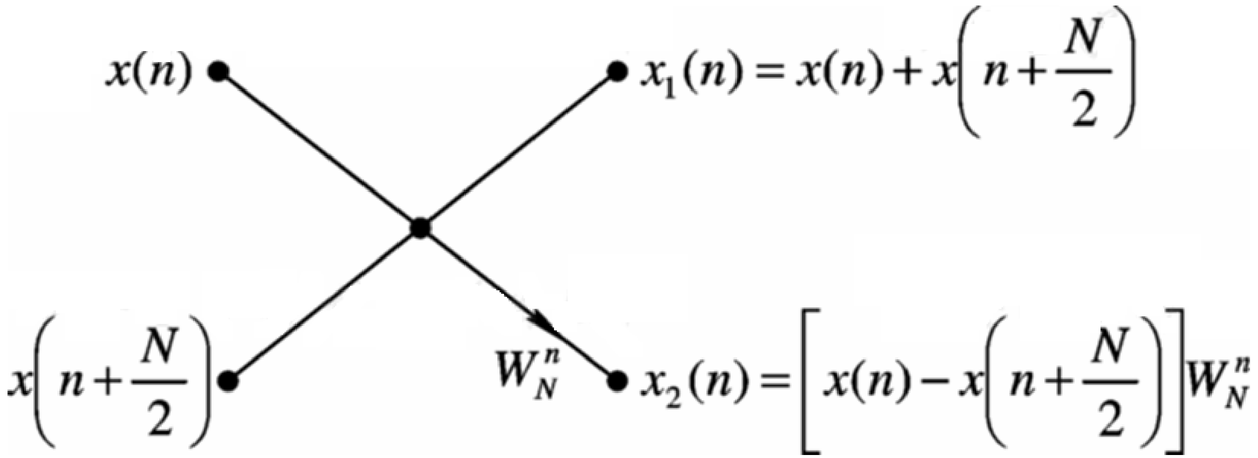


图4.2.10 DIF—FFT蝶形运算流图符号

4.2.5 频域抽取法FFT(DIF–FFT)

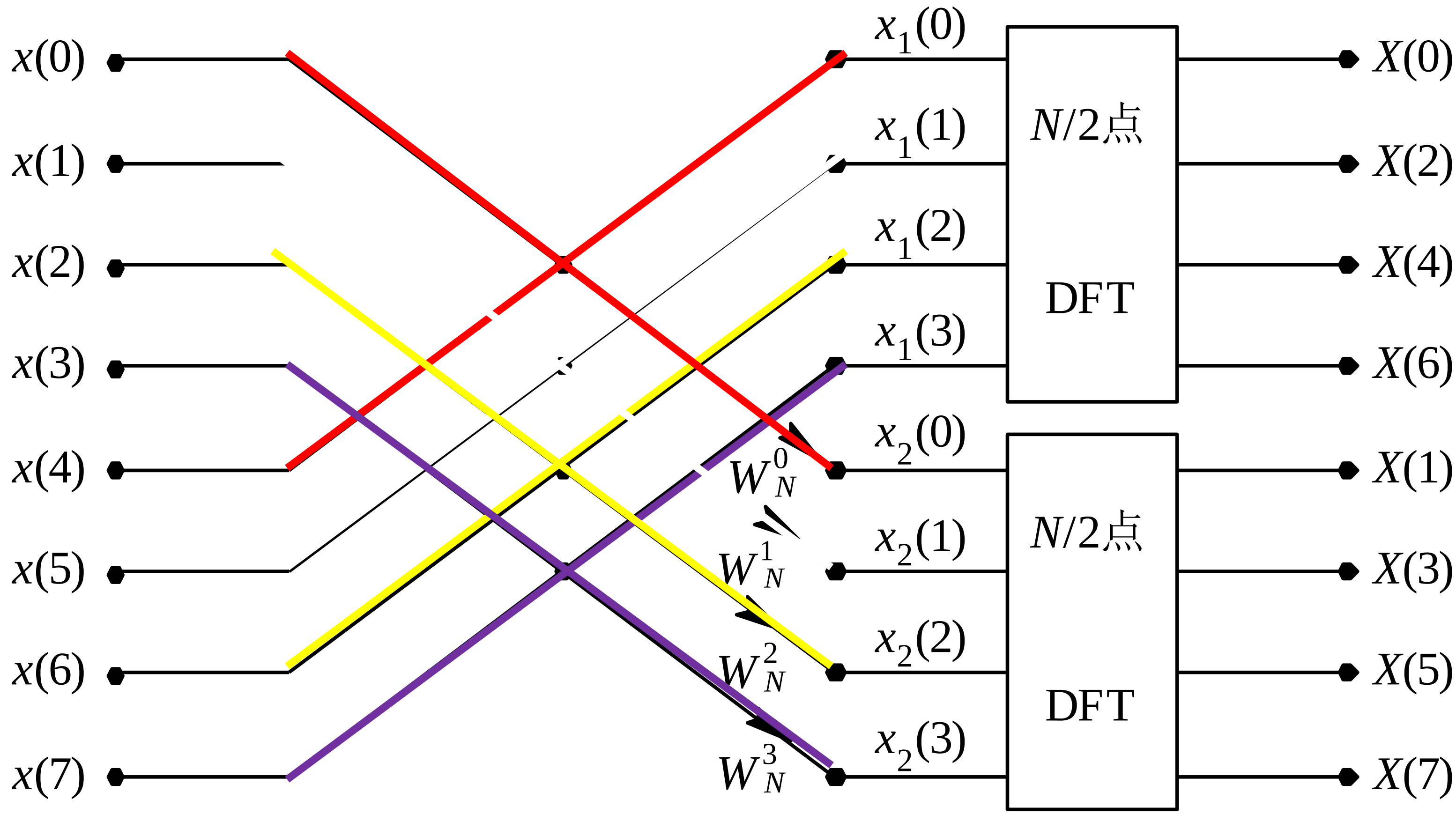


图4.2.11 DIF—FFT一次分解运算流图(N=8)

4.2.5 频域抽取法FFT(DIF-FFT)

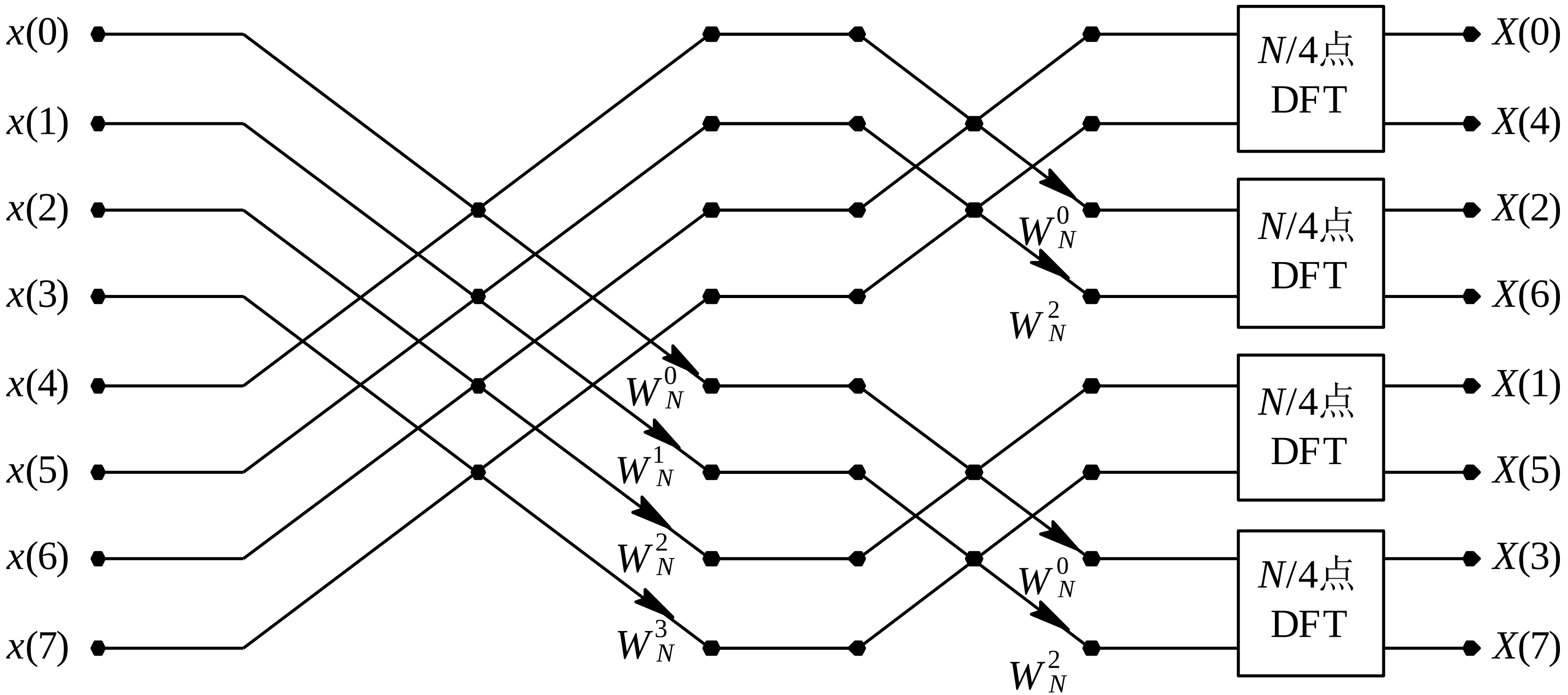


图4.2.12 DIF—FFT二次分解运算流图(N=8)

4.2.5 频域抽取法FFT(DIF-FFT)

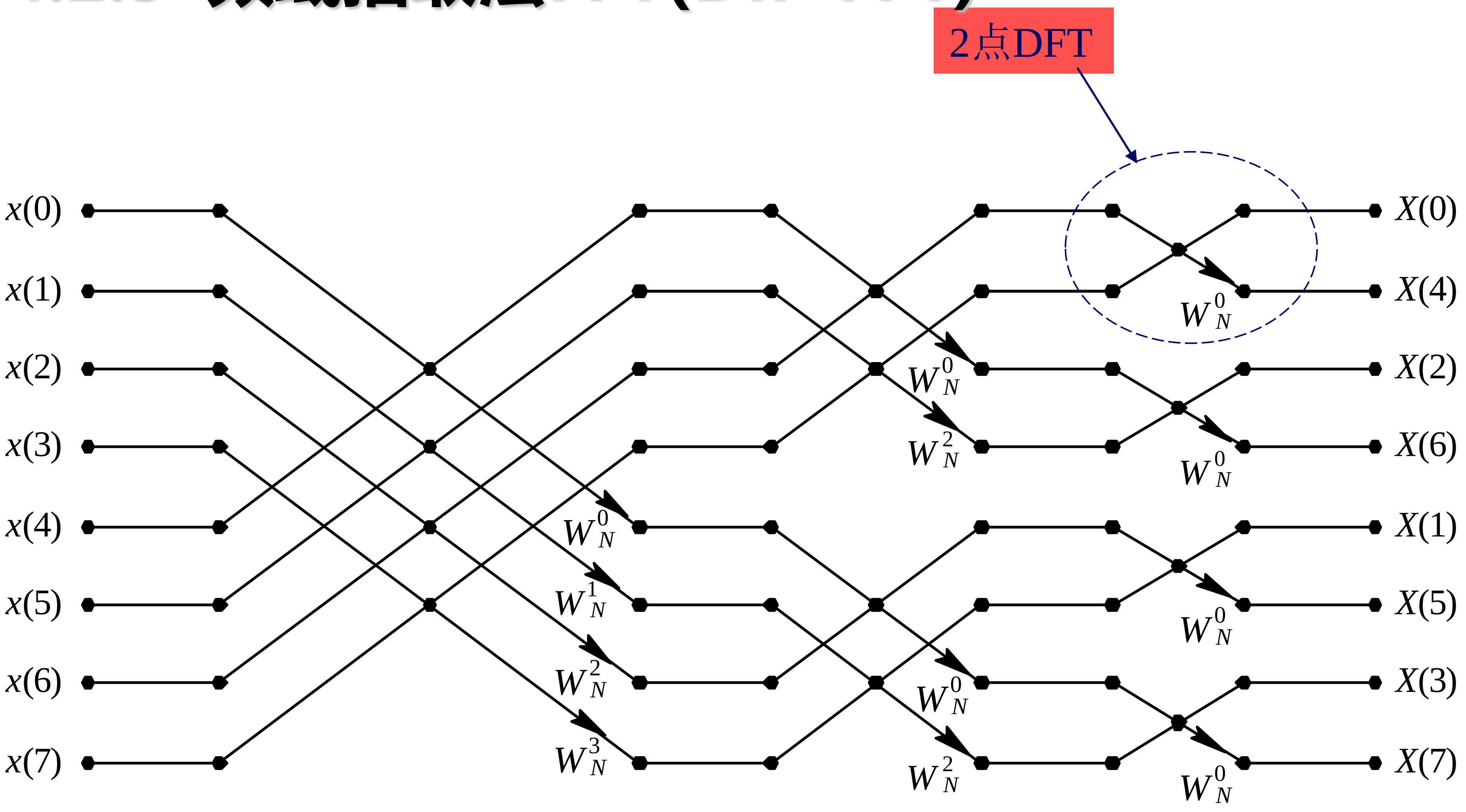


图4.2.13 DIF—FFT运算流图(N=8)

4.2.5 频域抽取法FFT(DIF–FFT)

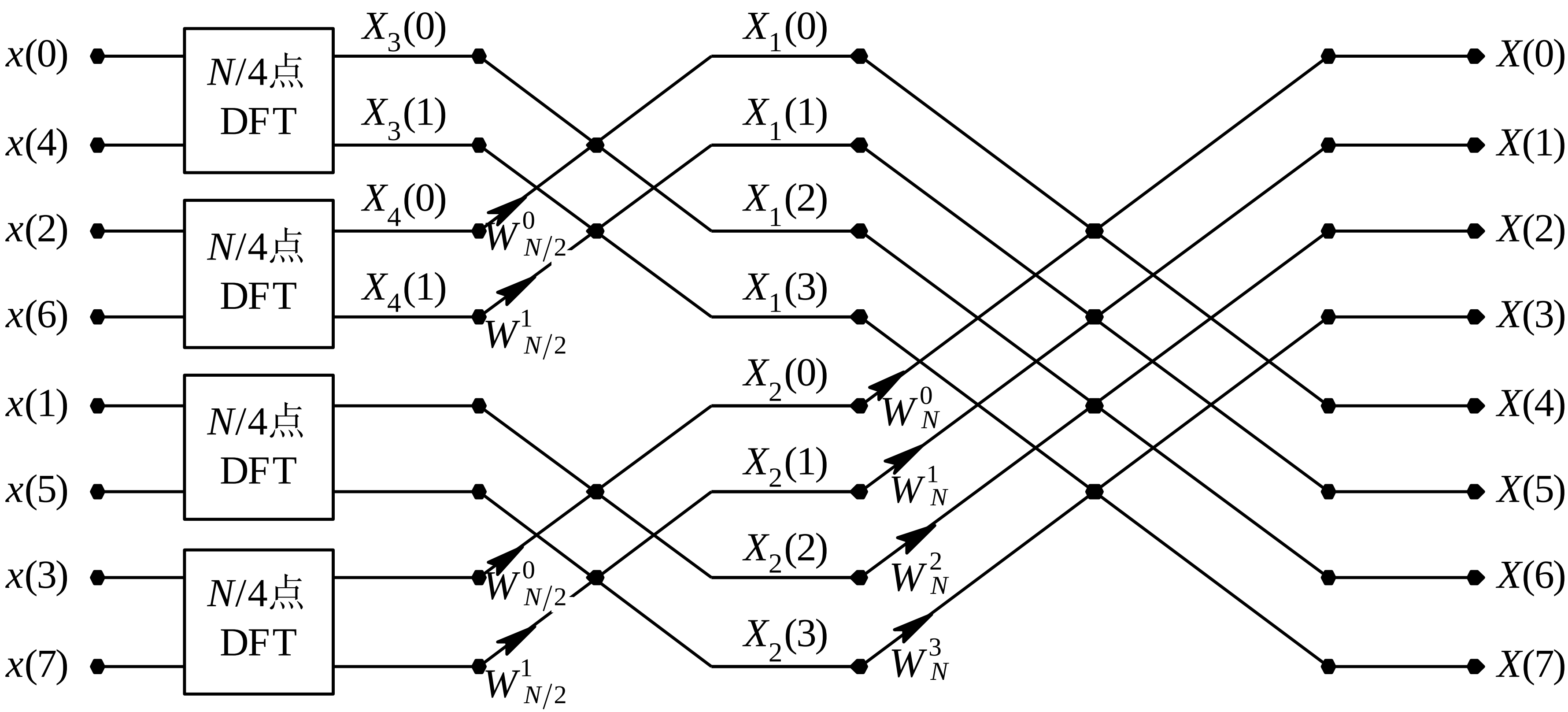
运算规律:

1. **原位计算**:运算过程无需增加新的存储单元

2. **旋转因子的变化规律**

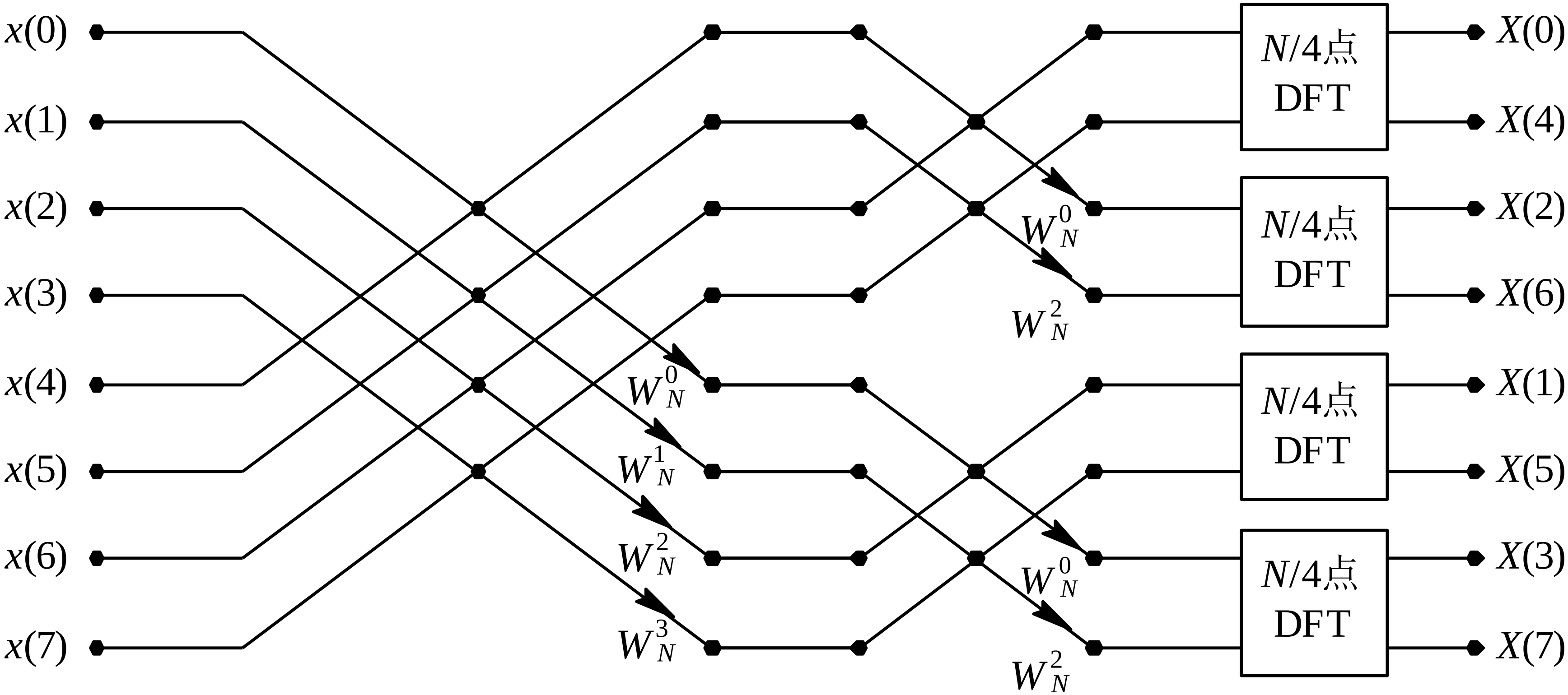
3. **输入序列的倒序**

4.2.5 DIT-FFT与DIF-FFT



DIT-FFT

4.2.5 DIT–FFT与DIF–FFT



DIF–FFT

4.2.5 频域抽取法FFT(DIF-FFT)

➢仔细观察基2DIF-FFT运算流图和基2DITFFT运算流图会发现，将频域抽取法的运算流图反转，并将输入变输出，输出变输入，正好得到时域抽取法的运算流图

➢按频域抽取算法与按时域抽取算法是两种等价的FFT算法，此外，在基2FFT的基础上，还有变形基2FFT运算流图，原理类似

4.2.6 IDFT的高效算法

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

IDFT: $$x(n)=IDFT[X(k)]=\frac{1}{N}\sum_{k=0}^{N-1}X(k)W_N^{-nk}$$

- 旋转因子指数变极性法
- 直接调用**FFT**子程序法

4.2.6 IDFT的高效算法

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

IDFT: $$x(n) = IDFT[X(k)] = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn} = \frac{1}{N} \left\{ DFT[X^*(k)]^* \right\}$$

4.2.6 IDFT的高效算法

■ 旋转因子指数变极性法

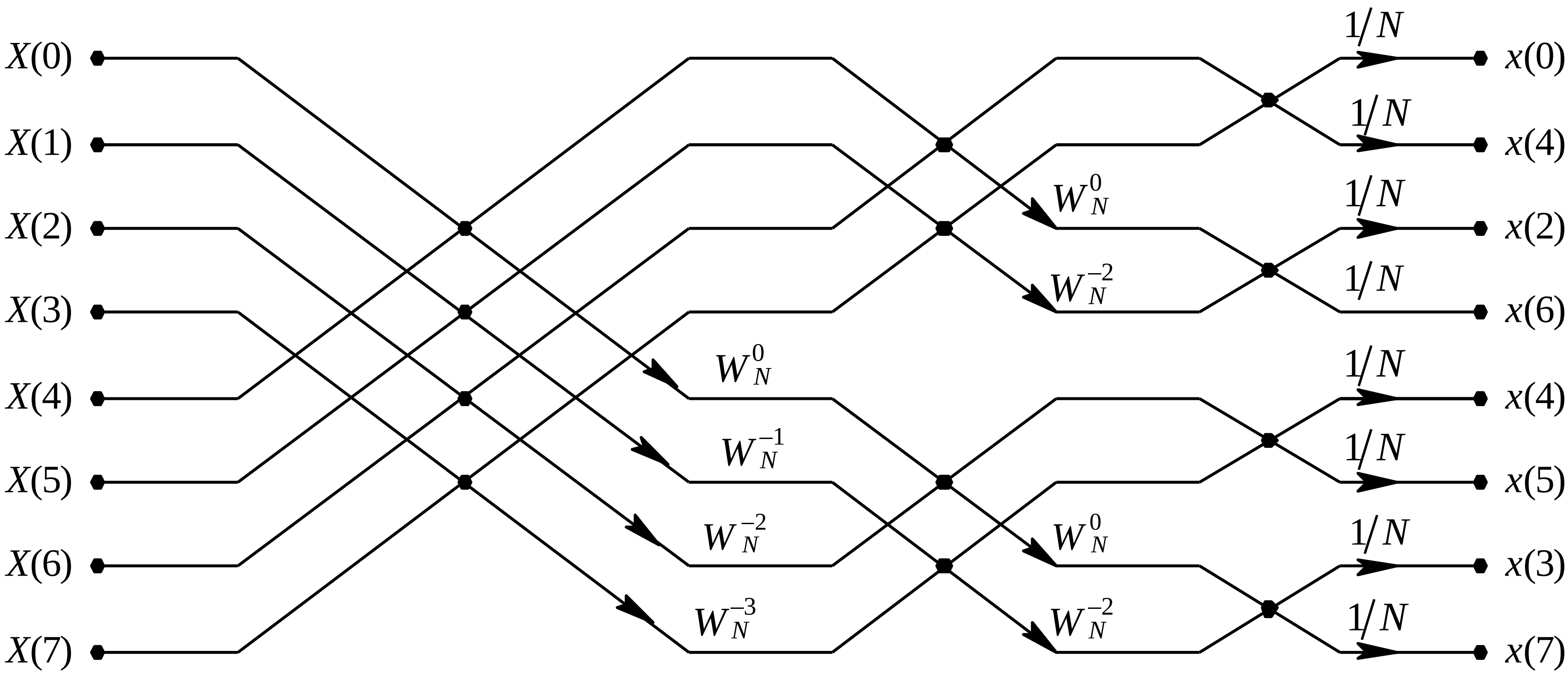


图4.2.16 DIT—IFFT运算流图

4.2.6 IDFT的高效算法

■ 旋转因子指数变极性法

乘法次数比图4.2.16
增加(M-1)N/2次

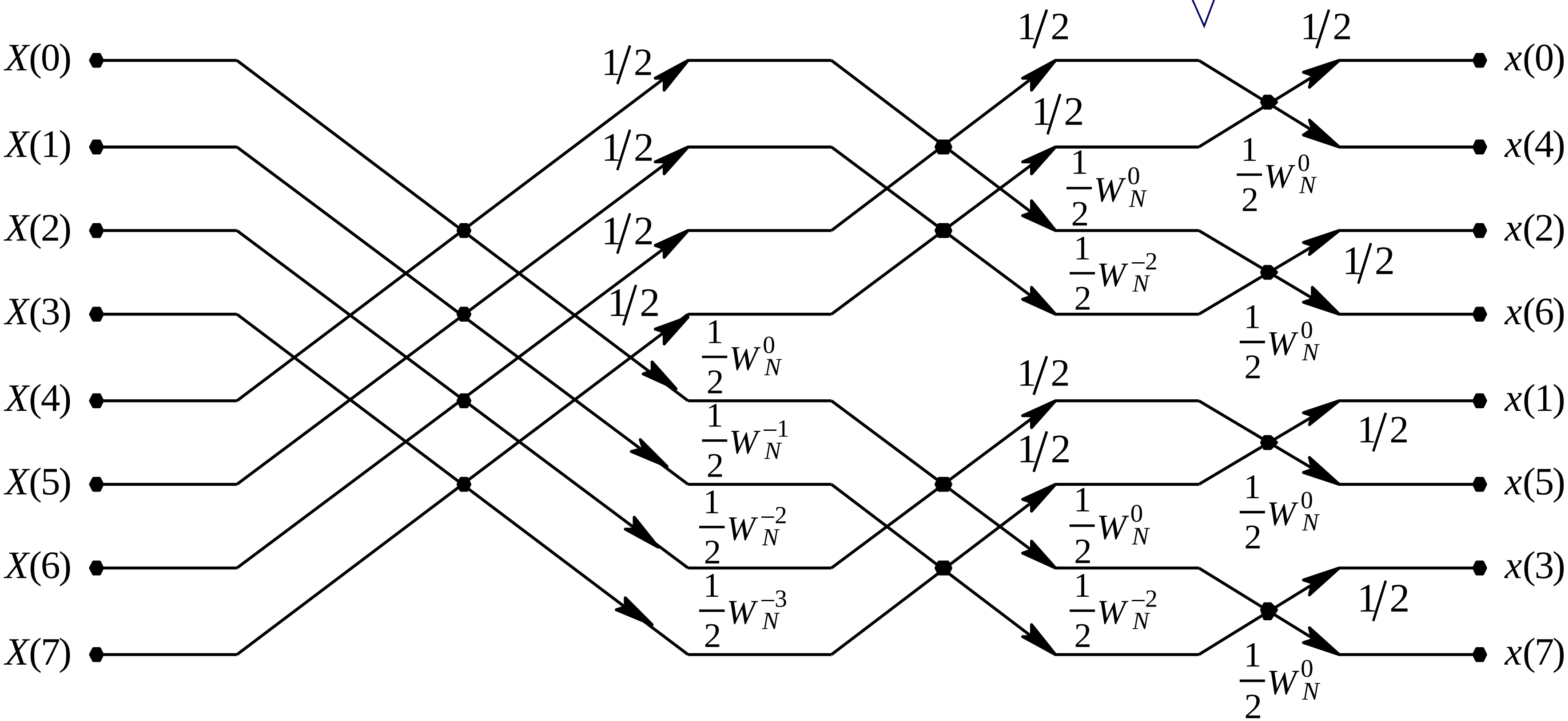


图4.2.17 DIT—IFFT运算流图(防止溢出)

4.2.6 IDFT的高效算法

直接调用FFT子程序法1

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

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

$$\Rightarrow\quad x(n)=\frac{1}{N}\left[\sum_{k=0}^{N-1}X^*(k)W_N^{kn}\right]^*=\frac{1}{N}\left\{DFT\left[X^*(k)\right]\right\}^*$$

Back

4.2.6 IDFT的高效算法

例：已知$X(k)$=｛-2，-2-2j，10，-2+2j｝，利用4点基-2 DIF-FFT算法流图计算$x(n)$。

解：

第一步，求出$X^*(k)$

$$X^*(k)=\{-2,\ -2+2j,\ 10,\ -2-2j\}$$

第二步，画出4点基-2 DIF-FFT算法流图。

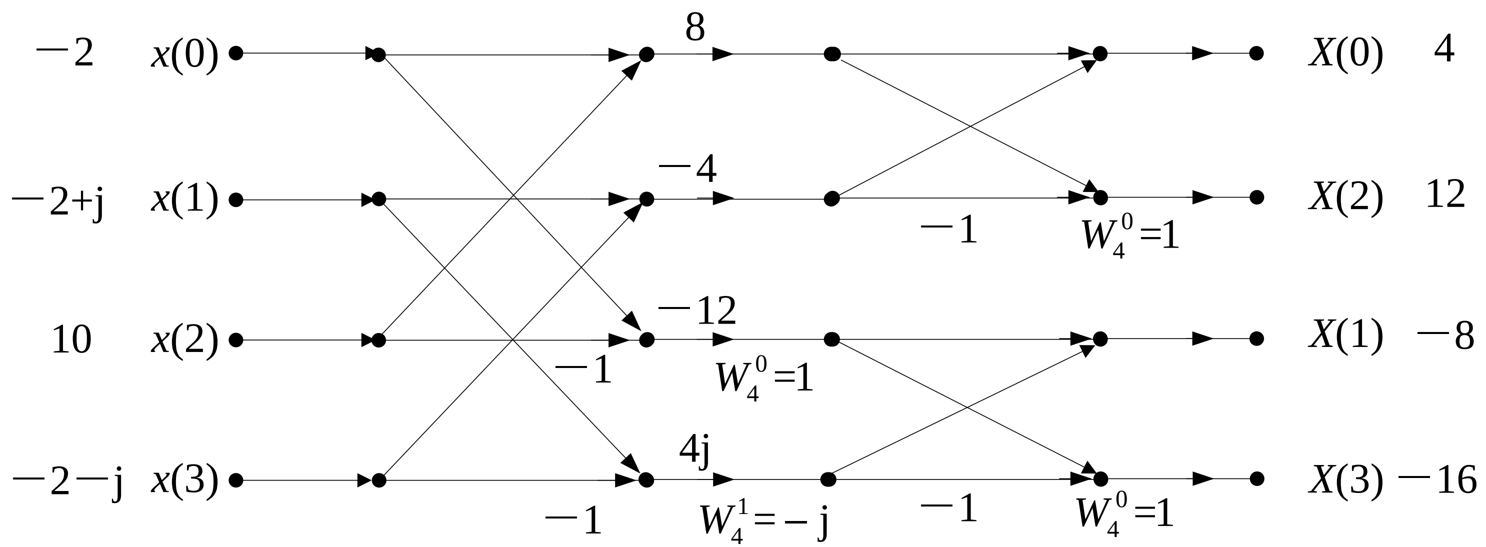
第三步，求出$x(n)$

$$x(n)=\frac{1}{4}\left\{\mathrm{DFT}[X^*(k)]\right\}^*=\frac{1}{4}\{4,-8,12,-16\}=\{1,-2,3,-4\}$$

4.2.6 IDFT的高效算法

例：已知$X(k)$=｛-2，-2-2j，10，-2+2j｝，利用4点基-2 DIF-FFT算法流图计算$x(n)$。

解：



4.3 进一步减少运算量的措施
4.3.1 多类蝶形单元运算

考虑旋转因子：

$$W_N^P = W_{N\cdot 2^{L-M}}^J = W_N^{J\cdot 2^{M-L}} \qquad J = 0,1,2,\cdots,2^{L-1}-1$$

$$p = J \cdot 2^{M-L}$$

注：在DFT中值为±1，±j的旋转因子称为无关紧要的旋转因子

例: N=2^3=8时的各级旋转因子表示如下：

$$W_N^p = W_{N/4}^J = W_{2^L}^J, J = 0$$

$$W_N^p = W_{N/2}^J = W_{2^L}^J, J = 0,1$$

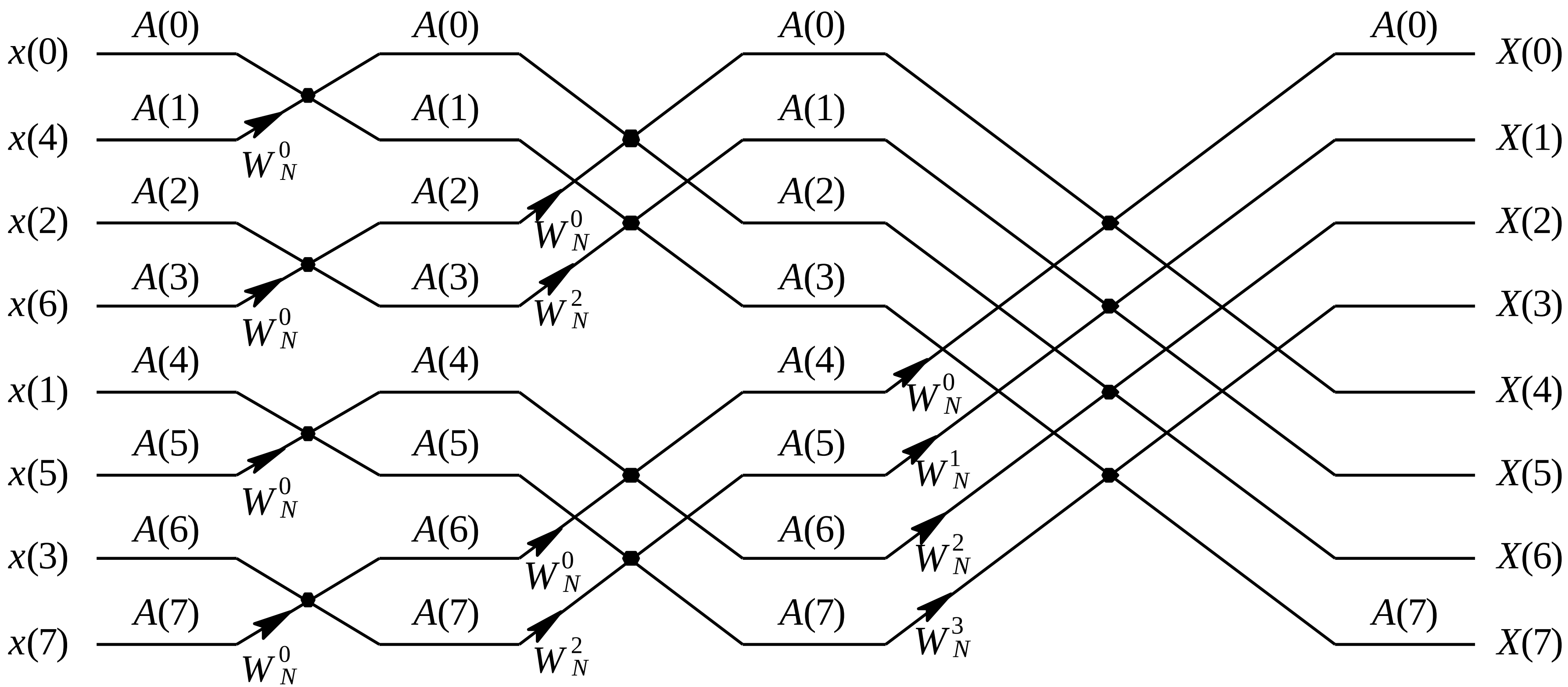
当L=1时，只有一种旋转因子$W_N^0 = 1$，所以第一级不需要乘法运算。

当L=2时，有两种旋转因子$W_N^0 = 1$ 及$W_N^{N/4} = -j$，所以第二级也不需要乘法运算。

4.3.1 多类蝶形单元运算

综上所述，除去第一、二两级后，所需复数乘法次数为

$$C_M = \frac{N}{2}(M-2) \tag{4.3.1}$$



N点DIT—FFT运算流图(N=8)

4.3.1 多类蝶形单元运算

每一级有$N/2$个蝶形运算单元，共2^{M-L}个蝶形；

每一级有2^{L-1}个不同的旋转因子，每个旋转因子对应$2^{M-L}=N/2^L$个蝶形运算

从$L=3$至$L=M$，每一级有两个无关紧要的旋转因子

$$W_N^0=1, \qquad W_N^{N/4}=-j$$

$$\sum_{L=3}^{M}\frac{N}{2^L}*2=2N\sum_{L=3}^{M}\left(\frac{1}{2}\right)^L=\frac{N}{2}-2 \qquad (4.3.2)$$

DIT—FFT的复乘次数：

$$C_M=\frac{N}{2}(M-2)-(\frac{N}{2}-2)=\frac{N}{2}(M-3)+2 \qquad (4.3.3)$$

4.3.1 多类蝶形单元运算

特殊复数运算

- 一般复乘：$(a+jb)*(c+jd)$ —— 4次实乘，2次实加；
- $W_N^{8/N}=(1-j)\sqrt{2}/2$ ：与任意复数相乘仅需2次实乘，2次实加；

$N=2^M$点DIT—FFT所需实数乘法次数：

$$R_M=4[\frac{N}{2}(M-3)+2]-2\sum_{L=3}^{M}\frac{N}{2^L}$$

$$=4[\frac{N}{2}(M-3)+2]-(\frac{N}{2}-2)$$

$$=N(2M-\frac{13}{2})+10 \tag{4.3.4}$$

蝶形运算分类：

一类蝶形单元运算：包含所有旋转因子；

二类蝶形单元运算：去掉$W^r_N = \pm 1$的旋转因子；

三类蝶形单元运算：再去掉$W^r_N = \pm j$的旋转因子；

四类蝶形单元运算：再处理$W^r_N = (1-j)\sqrt{2}/2$的旋转因子；

后三类称为多类蝶形单元运算。

蝶形运算单元越多，编程越复杂，乘法运算量越少。

例：三类蝶形运算中，乘法次数是一类蝶形运算的75%。

编程复杂度换取时间效率

4.3.2 旋转因子的生成

在FFT运算中，旋转因子 $W_N^m = cos(2\pi m/N) - jsin(2\pi m/N)$，求正弦和余弦函数值的计算量是很大的。

两种方法：

1. 运算中直接计算旋转因子；
2. 预先计算并保存，可提高运算速度，但耗内存；

4.3.3 实序列的FFT算法

三种情况：

1. 把实序列$x(n)$看作虚部为零的复序列，直接调用FFT； 时间浪费

2. 对两个N点的实序列$x(n)$和$h(n)$，构在复序列$y(n)$， 即

$$y(n)=x(n)+jh(n), n=0,1,\cdots,N-1$$

$$\left.\begin{aligned}X(k)&=DFT[x(n)]=Y_{ep}(k)=\frac{1}{2}[Y(k)+Y^*(N-k)]\\H(k)&=DFT[h(n)]=-jY_{op}(k)=\frac{1}{2j}[Y(k)-Y^*(N-k)]\end{aligned}\right\},k=0,1,\cdots,N-1$$

运算时间略多于一个N点FFT，可获得两个N点实序列的FFT

4.3.3 实序列的FFT算法

3. 设$x(n)$为N点实序列，取$x(n)$的偶数点和奇数点分别作为新构造序列$y(n)$的实部和虚部，即

$$x_1(n)=x(2n),x_2(n)=x(2n+1),n=0,1,\cdots,N/2-1$$
$$y(n)=x_1(n)+jx_2(n),n=0,1,\cdots,N/2-1$$

$$\left.\begin{aligned}X_1(k)&=DFT[x_1(n)]=Y_{ep}(k)\\X_2(k)&=DFT[x_2(n)]=-jY_{op}(k)\end{aligned}\right\},k=0,1,\cdots,\frac{N}{2}-1$$

4.3.3 实序列的FFT算法

根据DIT—FFT的思想，可得到

$$X(k)=X_1(k)+W_N^k X_2(k),\quad k=0,1,\cdots,N/2-1$$

由于$x(n)$为实序列，所以$X(k)$具有共轭对称性，X(k)的另外N/2点的值为

$$X(N-k)=X^*(k),k=1,2,\cdots,N/2-1$$

4.3.3 实序列的FFT算法

- 运算效率（复乘）：

$$\eta = \frac{M\dfrac{N}{2}}{M\dfrac{N}{4} + \dfrac{N}{2}} = \frac{2M}{M+2} \approx 2$$

Back

Part 4 作 业

提交作业：1、4、5