第 四 章

快速傅里叶变换

本章内容

4.1 引言

4.2 直接计算**DFT**的问题及改进途径

4.3 按时间抽取（**DIT**）的基**-2 FFT**算法（库利**-**图基算法）

4.4 按频率抽取（**DIF**）的基**-2 FFT**算法（桑德**-**图基算法）

4.5 离散傅里叶反变换的快速算法

4.6 线性卷积的**FFT**算法

4.1 引言

- 快速傅里叶变换FFT（Fast Fourier Transform）并不是一种新的变换，而是离散傅里叶变换变换（DFT）的一种快速算法。
- DFT在数字信号处理中非常有用。
- 在1965年前，DFT的计算量太大，没有获得真正的应用。
- 1965年库里（J.W.Cooley）和图基（J.W.Tukey）提出DFT的快速算法，随后，其它快速算法也相继提出，DFT的应用才真正开始。

4.2 直接计算DFT的问题及改进途径

- 有限长序列 x(n) 的DFT及IDFT为，

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

$$x(n)=\frac{1}{N}\sum_{k=0}^{N-1}X(k)W_N^{-nk},\qquad n=0,1,\mathrm{L}\ ,N-1$$

- DFT和IDFT区别:

(1) W_N的指数符号不同，

(2)常数因子（1/N）.

- 一般认为，两者的计算量完全一样.

1 DFT的计算量

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

$$x(n)=\frac{1}{N}\sum_{k=0}^{N-1}X(k)W_N^{-nk},\qquad n=0,1,\mathrm{L}\ ,N-1$$

- $x(n)$ 和 W_N^{nk} 都是复数，故计算一个 **DFT**

  (1) 需要N次复数乘法（$x(n)$ 和 W_N^{nk} 相乘）

  (2) 需要（N-1）次复数加法 。

- **求全部N个DFT值，需要的计算量为：**

  复数乘法： N^2

  复数加法： $N(N-1)$

- 复数乘法

$$x(n)\bullet W_N^{nk}=\left\{\mathrm{Re}\left[x(n)\right]+j\,\mathrm{Im}\left[x(n)\right]\right\}\bullet\left\{\mathrm{Re}\left[W_N^{nk}\right]+j\,\mathrm{Im}\left[W_N^{nk}\right]\right\}$$

$$=\mathrm{Re}\left[x(n)\right]\mathrm{Re}\left[W_N^{nk}\right]-\mathrm{Im}\left[x(n)\right]\mathrm{Im}\left[W_N^{nk}\right]$$

$$+j\left\{\mathrm{Re}\left[x(n)\right]\mathrm{Im}\left[W_N^{nk}\right]+\mathrm{Im}\left[x(n)\right]\mathrm{Re}\left[W_N^{nk}\right]\right\}$$

1次复数乘法 = 4次实数乘法＋2次实数加法

- 复数的加法

$$x(n)+W_N^{nk}=\mathrm{Re}\left[x(n)\right]+\mathrm{Re}\left[W_N^{nk}\right]+j\left\{\mathrm{Im}\left[x(n)\right]+\mathrm{Im}\left[W_N^{nk}\right]\right.$$

1次复数加法 = 2次实数加法

- 一个X(k)（k固定）的实数计算量

乘法次数：4N

加法次数：$2N+2(N-1)=2(2N-1)$

- N点DFT的实数计算量

乘法次数：$4N^2$

加法次数：$N\times 2(2N-1)=2N(2N-1)$

- 总结：N点DFT的复数计算量

乘法次数：N^2

加法次数：$N(N-1)$

- 注意： 实际上有些值可以不要计算，如

$$W_N^0 = 1,\ W_N^{N/2} = -1,\ W_N^{N/4} = -j$$

就不用计算了。

- 因此，实际计算量要少一点。但当N很大时，其所占比重是很小的。
- 故在分析计算量时，一般不考虑其影响。
- 例如：**N＝1024，DFT复乘：N^2>100万次，**计算量很大，不适合实时信号处理。
- 如何减少**DFT**的计算量？

2 减少DFT计算量的途径

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

分析 DFT公式，x(n)不能随意改变，只能从 W_N^{nk} 想办法。

分析系数 W_N^{nk} ，特性如下

（1）对称性 $\left(W_N^{nk}\right)^*=W_N^{-nk}$

（2）周期性 $W_N^{nk}=W_N^{(n+N)k}=W_N^{n(k+N)}$

（3）可约性 $W_N^{nk}=W_{mN}^{mnk}=W_{N/m}^{nN/m}$

故有，$W_N^{n(N-k)} = W_N^{(N-n)k} = W_N^{-nk}$ L

减少DFT计算量的方法：

（1）利用 W_N^{nk} 的特性，使DFT中的某些项可以合并；

$$(ax+bx) = (a+b)x$$

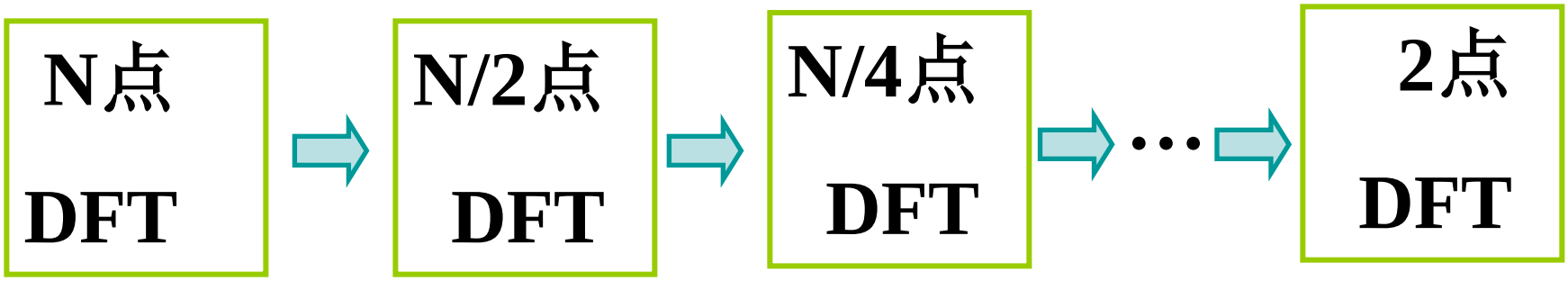
2次乘法、1次加法 → 2次乘法、1次加法

（2）利用 W_N^{nk} 的特性，将长序列的DFT分解为短序列的DFT。

复数乘法N^2，N越小，计算量越小

第二点比第一点更为重要！

FFT的核心思想是：



DFT的复数乘法计算量：N^2

N越小，计算量越小！

快速傅里叶变换（FFT）即由此产生！

前提：DFT计算结果正确

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

- 问题: 如何分解最有效?
  - 可以对时间变量 **n** 分解
  - 可以对频率变量 **k** 分解

两类**FFT**算法：

（1）时间抽取法 (decimation-in-time, DIT)

（2）频率抽取法 (decimation-in-frequency, DIF)

4.3 按时间抽取（DIT）的基-2 FFT算法（库里-图基法）

一、算法原理

二、运算量

三、按时间抽取的FFT特点

四、按时间抽取的FFT算法的其它形式

一、算法原理

- 序列 $x(n)$ 的长度为： $N=2^L$ ， **L为正整数**
- **如果不满足要求，则补零。**
- **N为2的整数幂的FFT称作基-2 FFT。**

时间抽取法就是将 $N=2^L$ **的序列** $x(n)$ **按照** n **的奇(odd)偶(even)分成两部分：**

$$\left.\begin{aligned} x(2r) &= x_1(r) \\ x(2r+1) &= x_2(r) \end{aligned}\right\}, \qquad r = 0, 1, \mathrm{L}\ , \frac{N}{2}-1$$

$$x(n) = x(2r) + x(2r+1)$$

问题：可否存在： $x(n) = x_1(r) + x_2(r)$ ？

x(n)的DFT为，

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

$$=\sum_{\substack{n=0\\ n\text{为偶数}}}^{N-1}x(n)W_N^{nk}+\sum_{\substack{n=0\\ n\text{为奇数}}}^{N-1}x(n)W_N^{nk}$$

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

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

$$=X_1(k)+W_N^kX_2(k)\ ,\quad k=0,1,2,\mathrm{L}\ ,N-1$$

式中，$X_1(k)$ 和 $X_2(k)$ 分别是 $x_1(r)$ 和 $x_2(r)$ 的N/2点DFT

$$X_1(k)=\sum_{r=0}^{\frac{N}{2}-1}x_1(r)W_{N/2}^{rk}=\sum_{r=0}^{\frac{N}{2}-1}x(2r)W_{N/2}^{rk}$$

$$k=0,1,\mathrm{L}\ ,\frac{N}{2}-1$$

$$X_2(k)=\sum_{r=0}^{\frac{N}{2}-1}x_2(r)W_{N/2}^{rk}=\sum_{r=0}^{\frac{N}{2}-1}x(2r+1)W_{N/2}^{rk}$$

结论：一个N点DFT可以分解为两个N/2点的DFT。

问题：由N/2点的DFT计算的N点的DFT只能获得k=0,1,…,N/2-1点X(k)，即前半部分？

- 后半部分如何求？即如何由$X_1(k)$和$X_2(k)$来求全部的 $X(k)$ ？

- 利用系数 W_N^{nk} 的周期性： $W_{N/2}^{rk} = W_{N/2}^{r(k+\frac{N}{2})}$

得到，

$$X_1(\frac{N}{2}+k) = \sum_{r=0}^{\frac{N}{2}-1} x_1(r) W_{N/2}^{r(k+\frac{N}{2})}$$

$$= \sum_{r=0}^{\frac{N}{2}-1} x_1(r) W_{N/2}^{rk} = X_1(k) \qquad k = 0,1,\mathrm{L}\ ,\frac{N}{2}-1$$

同理，

$$X_2(\frac{N}{2}+k) = X_2(k) \qquad k = 0,1,\mathrm{L}\ ,\frac{N}{2}-1$$

• 结论：后半部分k值所对应的 $X_1(k)$ 和 $X_2(k)$ 分别和前半部分k值所对应的 $X_1(k)$ 和 $X_2(k)$ 相等。

• $X_1(k)$ 和 $X_2(k)$ 是N/2点的DFT，其隐含周期性，周期为N/2。

$$X(k)=X_1(k)+W_N^k X_2(k)\ ,\quad k=0,1,2,\mathrm{L}\ ,N-1$$

考虑 $$W_N^{(\frac{N}{2}+k)}=W_N^{\frac{N}{2}}W_N^k=-W_N^k$$

故有，
$$X(\frac{N}{2}+k)=X_1(\frac{N}{2}+k)+W_N^{(\frac{N}{2}+k)}X_2(\frac{N}{2}+k)$$

$$=X_1(k)-W_N^k X_2(k)\qquad k=0,1,\mathrm{L}\ ,\frac{N}{2}-1$$

- 综合：

前半部分 $X(k)$ ，

$$X(k)=X_1(k)+W_N^kX_2(k) \qquad k=0,1,\mathrm{L}\ ,\frac{N}{2}-1$$

后半部分 $X(k)$ ，

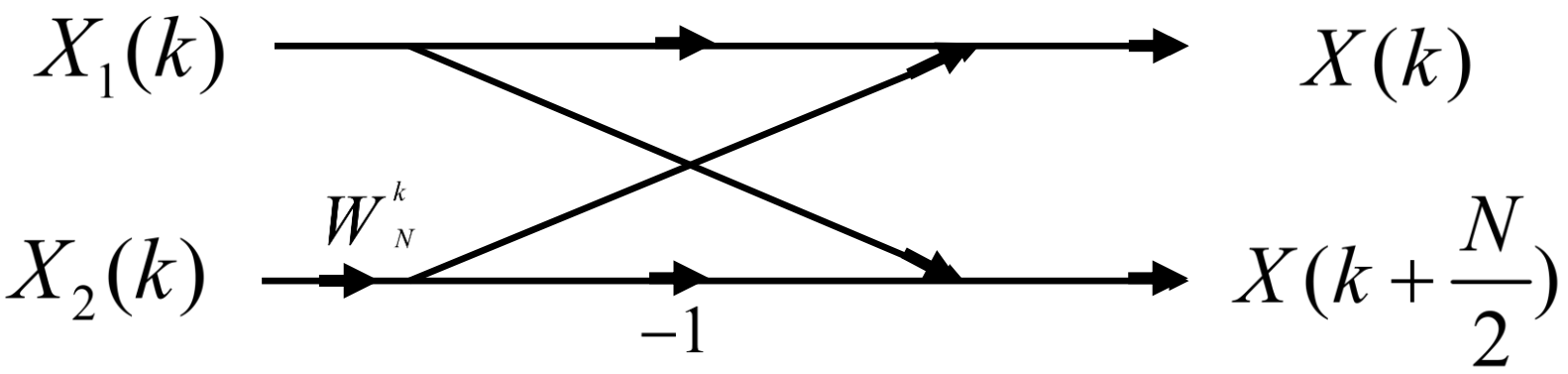
$$X(k+\frac{N}{2})=X_1(k)-W_N^kX_2(k) \quad k=0,1,\mathrm{L}\ ,\frac{N}{2}-1$$

$$X(k)=X_1(k)+W_N^k X_2(k)$$

$$X(k+\frac{N}{2})=X_1(k)-W_N^k X_2(k)$$

$$k=0,1,\mathrm{L}\ ,\frac{N}{2}-1$$

- 采用蝶形(butterfly)运算！



- 蝶形运算的计算量

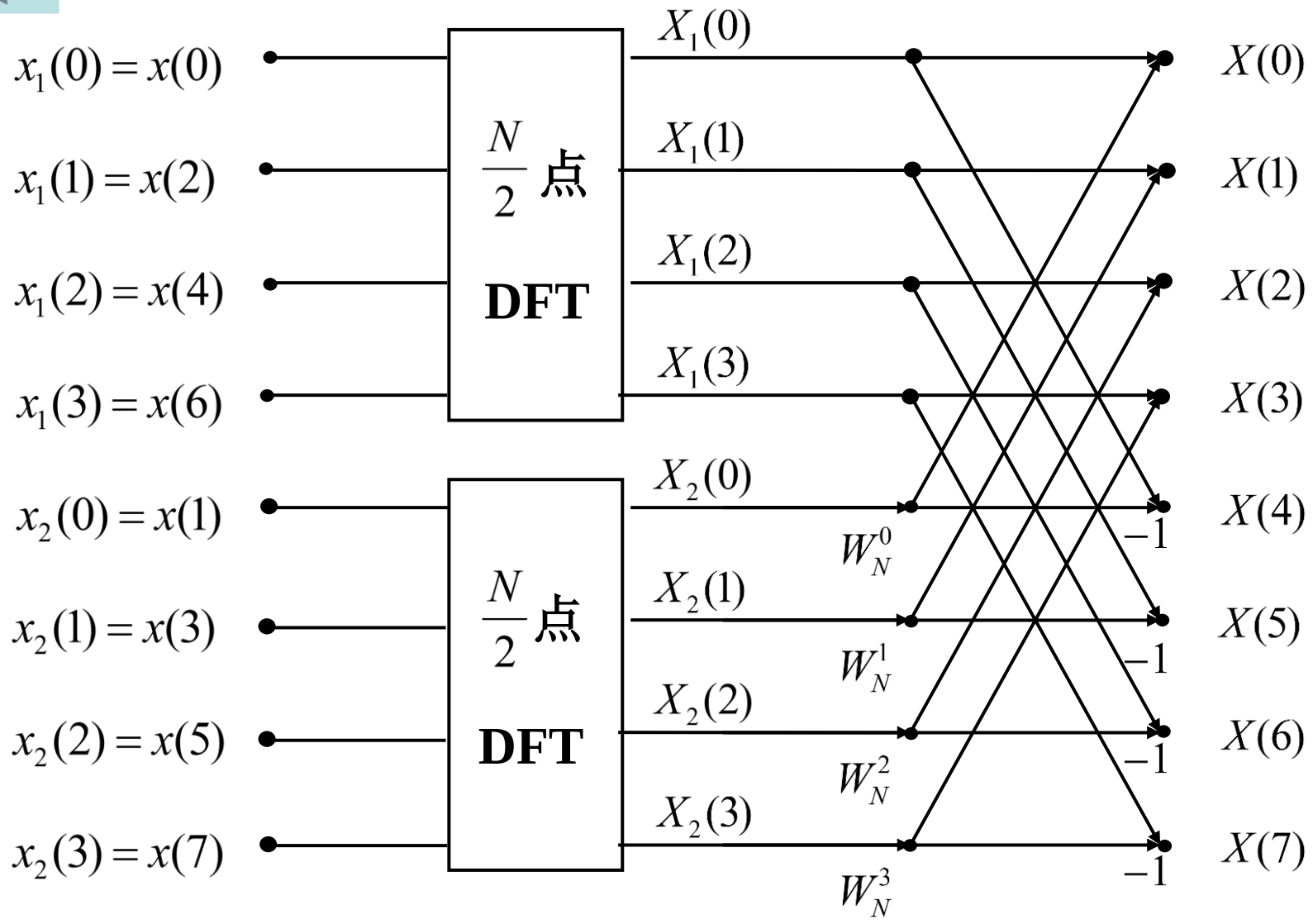
复数乘法： 1次

复数加法： 2次

• 按照时间抽取方法，将一个N点DFT分解为两个N/2点的DFT的分解过程如图所示。

（ 假设N=8）

（将k=0,1,2,3代入到公式中）



按时间抽取，一个N点DFT分解为两个N/2点DFT

分解后的DFT计算量分析

- 假设直接计算N/2点的DFT，其计算量为

复数乘法：$\left(\frac{N}{2}\right)^2$

复数加法：$\frac{N}{2}\left(\frac{N}{2}-1\right)$

- 两个N/2点的DFT计算量

复数乘法：$2\times\left(\frac{N}{2}\right)^2=\frac{N^2}{2}$

复数加法：$2\times\frac{N}{2}\left(\frac{N}{2}-1\right)=N\left(\frac{N}{2}-1\right)$

- 一个蝶形运算的计算量

  复数乘法： 1 次

  复数加法： 2 次

- N/2个蝶形计算量

  复数乘法： N/2 次

  复数加法： N 次

- **N点DFT所需的计算量（采用DIT分解）**

复数乘法：$\frac{N^2}{2}+\frac{N}{2}=\frac{N(N+1)}{2}\approx\frac{N^2}{2}$

复数加法：$N(\frac{N}{2}-1)+N=\frac{N^2}{2}$

比较N点DFT计算量（直接计算）

复数乘法：N^2　　复数加法：$N(N-1)\approx N^2$

计算量（乘法、加法）差不多减少一半！

由于 $N=2^L$ ，N/2仍然是偶数，可以进一步将N/2点 $x_1(r)$ 的DFT分解为两个N/4点的DFT。

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

$$X_1(k)=\sum_{l=0}^{\frac{N}{4}-1}x_1(2l)W_{N/2}^{2lk}+\sum_{l=0}^{\frac{N}{4}-1}x_1(2l+1)W_{N/2}^{(2l+1)k}$$

$$=\sum_{l=0}^{\frac{N}{4}-1}x_3(l)W_{N/4}^{lk}+W_{N/2}^{k}\sum_{l=0}^{\frac{N}{4}-1}x_4(l)W_{N/4}^{lk}$$

$$=X_3(k)+W_{N/2}^{k}X_4(k), \qquad k=0,1,\mathrm{L}\ ,\frac{N}{4}-1$$

式中，$X_3(k)$和$X_4(k)$分别是$x_3(l)$和$x_4(l)$的N/4点DFT

$$X_3(k)=\sum_{l=0}^{\frac{N}{4}-1}x_3(l)W_{N/4}^{rk}$$

$$X_4(k)=\sum_{l=0}^{\frac{N}{4}-1}x_4(l)W_{N/4}^{rk}$$

上式计算的是 $X_1(k)$ 的前 N/4 点的FFT

同样有 $X_1(k)$ 的后N/4点FFT：

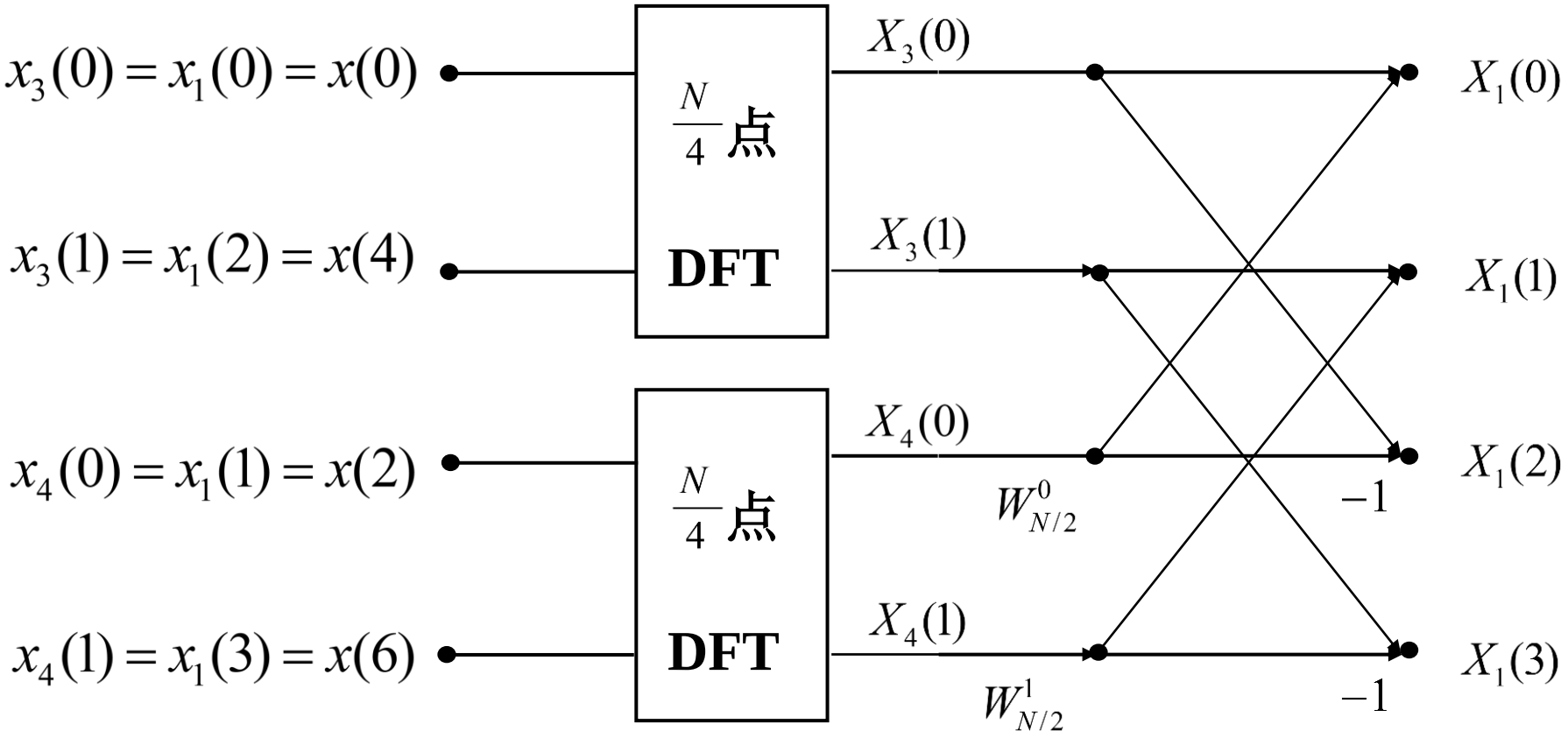
$$X_1(k+\frac{N}{4})=X_3(k)-W_{N/2}^{k}X_4(k),\qquad k=0,1,\mathrm{L}\ ,\frac{N}{4}-1$$

同理，可以对 $x_2(r)$ 进行分解！

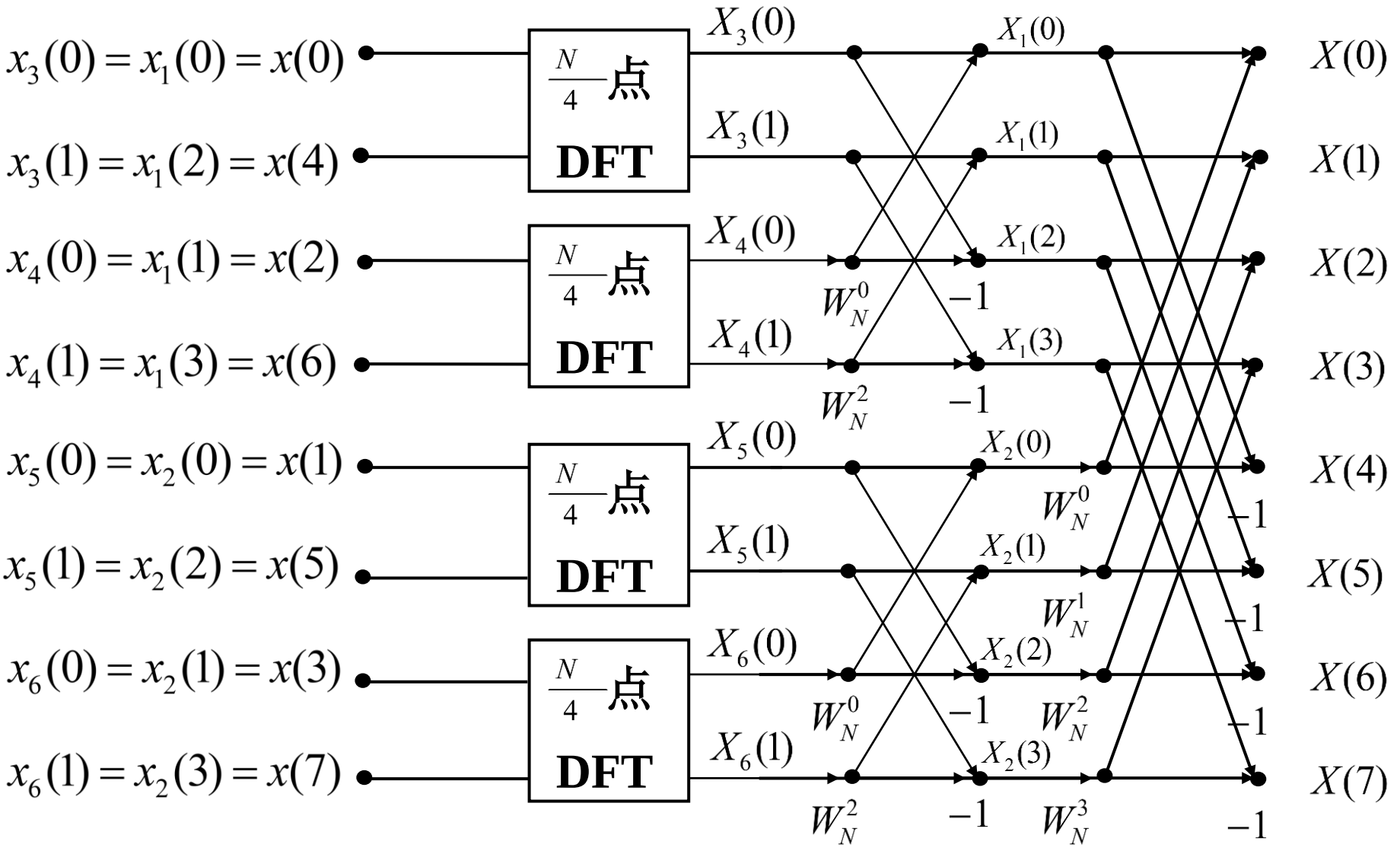
$$\left.\begin{aligned} X_2(k) &= X_5(k) + W_{N/2}^{k} X_6(k) \\ X_2(k+\frac{N}{4}) &= X_5(k) - W_{N/2}^{k} X_6(k) \end{aligned}\right\} \quad k = 0, 1, \mathrm{L}\ , \frac{N}{4} - 1$$

$$X_5(k) = \sum_{l=0}^{\frac{N}{4}-1} x_2(2l) W_{N/4}^{rk} = \sum_{l=0}^{\frac{N}{4}-1} x_5(l) W_{N/4}^{rk}$$

$$X_6(k) = \sum_{l=0}^{\frac{N}{4}-1} x_2(2l+1) W_{N/4}^{rk} = \sum_{l=0}^{\frac{N}{4}-1} x_6(l) W_{N/4}^{rk}$$



一个N/2点DFT分解为两个N/4点DFT



按时间抽取，一个N点DFT分解为四个N/4点DFT

- 考虑分解过程中，序列标号的变化！

- 第一次分解：（以N=8为例）

偶序列　　奇序列

$$x(2r)=x_1(r)$$

$$x(2r+1)=x_2(r)$$

$$r=0,1,2,3$$

r	0	1	2	3
$n=2r$	0	2	4	6

r	0	1	2	3
$n=2r+1$	1	3	5	7

$$x_1(0)=x(0),\ x_1(1)=x(2)$$

$$x_1(2)=x(4),\ x_1(3)=x(6)$$

$$x_2(0)=x(1),\ x_2(1)=x(3)$$

$$x_2(2)=x(5),\ x_2(3)=x(7)$$

- 第二次分解：

偶序列中的偶序列

$$x_1(2l) = x_3(l)$$

偶序列中的奇序列

$$x_1(2l+1) = x_4(l)$$

$$l = 0, 1$$

l	0	1
$r = 2l$	0	2
$n = 2r$	0	4

l	0	1
$r = 2l+1$	1	3
$n = 2r$	2	6

$$x_3(0) = x_1(0) = x(0)$$
$$x_3(1) = x_1(2) = x(4)$$

$$x_4(0) = x_1(1) = x(2)$$
$$x_4(1) = x_1(3) = x(6)$$

奇序列中的偶序列　　奇序列中的奇序列

$$x_2(2l) = x_5(l)$$

$$x_2(2l+1) = x_6(l)$$

$$l = 0, 1$$

l	0	1
$r = 2l$	0	2
$n = 2r+1$	1	5

l	0	1
$r = 2l+1$	1	3
$n = 2r+1$	3	7

$$x_5(0) = x_2(0) = x(1)$$

$$x_5(1) = x_2(2) = x(5)$$

$$x_6(0) = x_2(1) = x(3)$$

$$x_6(1) = x_2(3) = x(7)$$

- **第三次分解：2点DFT**

其输出为 $X_3(k), X_4(k), X_5(k), X_6(k),\ k=0,1$

此时也是一个蝶形运算，例如

$$X_4(0)=x_4(0)+W_2^0 x_4(1)=x(2)+x(6)$$

$$X_4(1)=x_4(0)-W_2^0 x_4(1)=x(2)-x(6)$$

但实际上，由于 $W_2^0=1$ ，

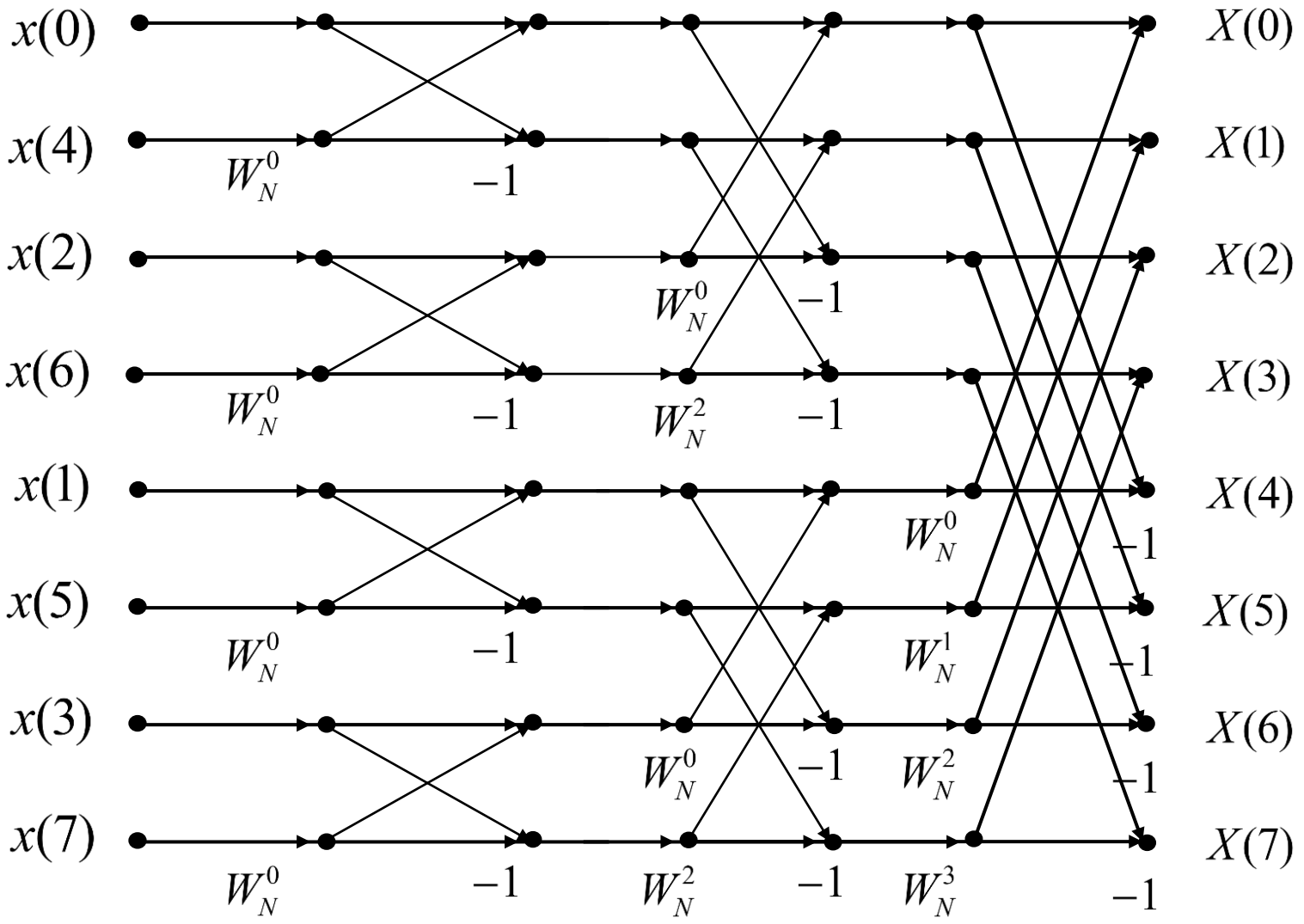
故不必进行蝶型运算, 直接加减即可。

- “按时间抽选法”名称的由来

以上方法每一步都是按照输入序列在时间上的次序是属于奇数还是偶数来分解（抽取），因此称作“按时间抽选法”

- N点FFT $N=2^L$ 例如：N=8
- L级FFT L=3
- 每一级有N/2个蝶形运算 N/2=4
- 每一个蝶形运算是一样的

4点FFT如何画？



N=8按时间抽取FFT运算流图

二、运算量

1. 时间抽取法FFT，当 $N=2^L$，有L级（列）蝶形，每一级有N/2个蝶形运算。

2. 每一个蝶形运算的计算量：

复数乘法： 1 复数加法： 2

3. 每一级蝶形运算的计算量：

复数乘法： N/2 复数加法： N

4. 全部蝶形运算的计算量：

复数乘法：$\frac{N}{2}L=\frac{N}{2}\log_2 N$ 复数加法：$NL=N\log_2 N$

5. 实际上，有一些不必计算，如 $W_N^0 = 1, W_N^{N/4} = -j$

6. 数量：$W_N^0 = 1,$ **N-1个** $W_N^{N/4} = -j$ **N/2个**

7. 直接计算**DFT**的计算量

复数乘法： N^2 复数加法： $N(N-1)$

8. 两者之比为（复数乘法）：

$$\frac{N^2}{\frac{N}{2}\log_2 N} = \frac{2N}{\log_2 N}$$

表4-1 FFT算法与直接DFT算法的乘法次数比较

N	N^2	$\frac{N}{2}\log_2 N$	$\frac{2N}{\log_2 N}$
2	4	1	4.0
4	16	4	4.0
8	64	12	5.4
16	256	32	8.0
32	1024	80	12.8
64	4096	192	21.4
128	16384	448	36.6
256	65536	1024	64.0
512	262144	2304	113.8
1024	1048576	5120	204.8
2048	4194304	11264	372.4

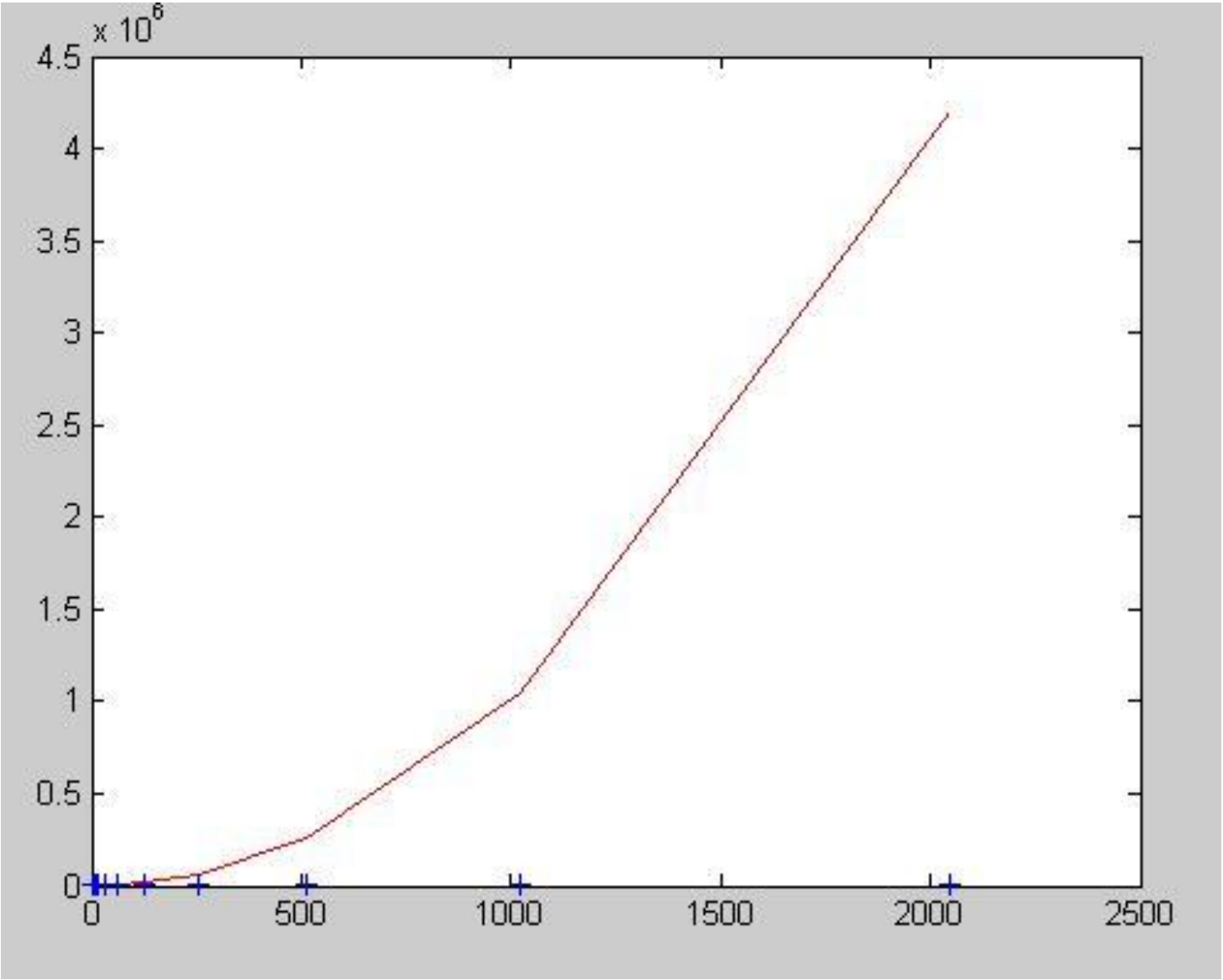
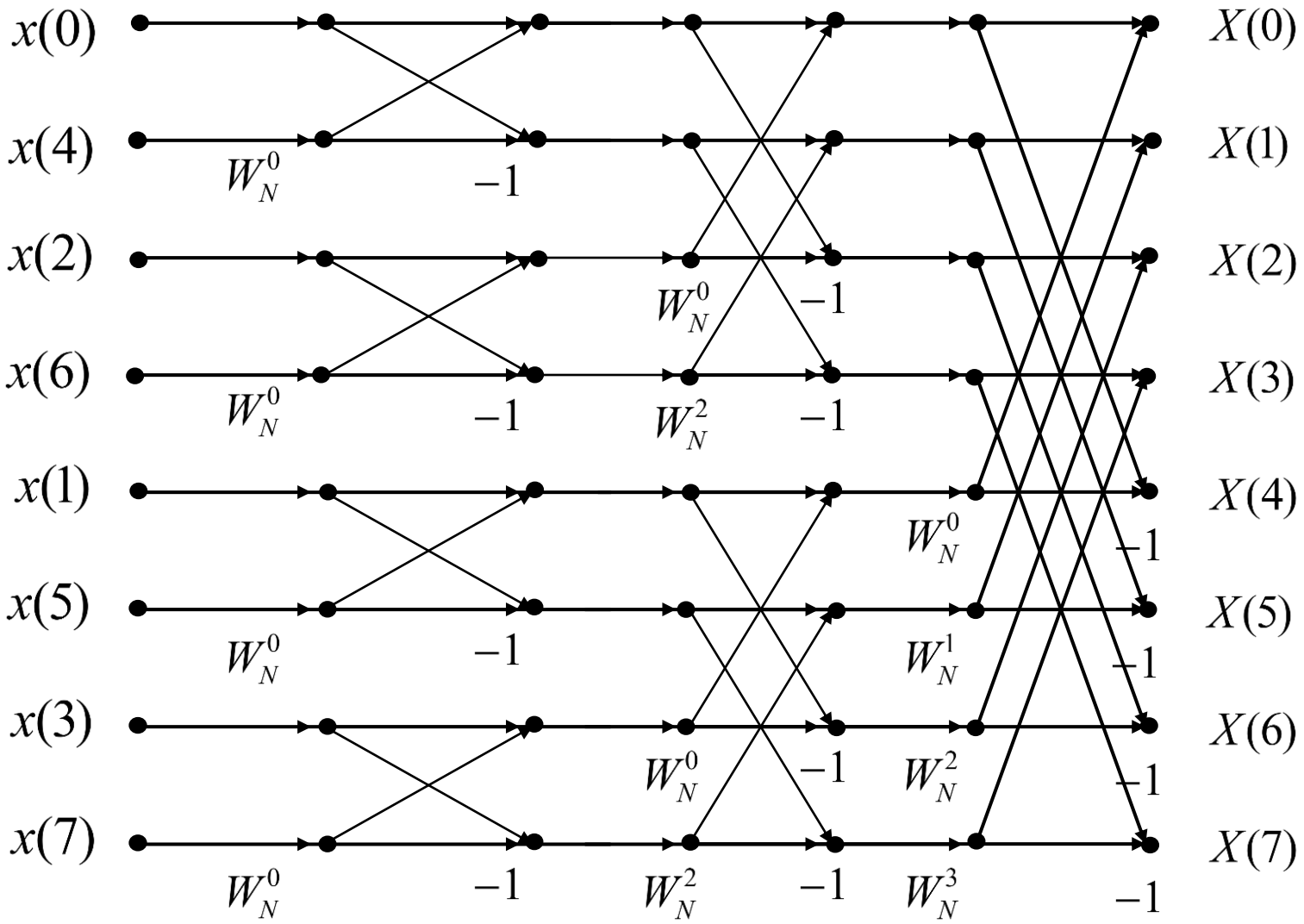


图4-6 FFT算法与直接DFT算法的比较

二、按时间抽取FFT的特点

1. 原位运算（同址运算）
2. 倒位序规则
3. 倒位序的实现
4. 蝶形运算两结点之间的距离
5. 系数 W_N^r 的确定
6. 存储单元



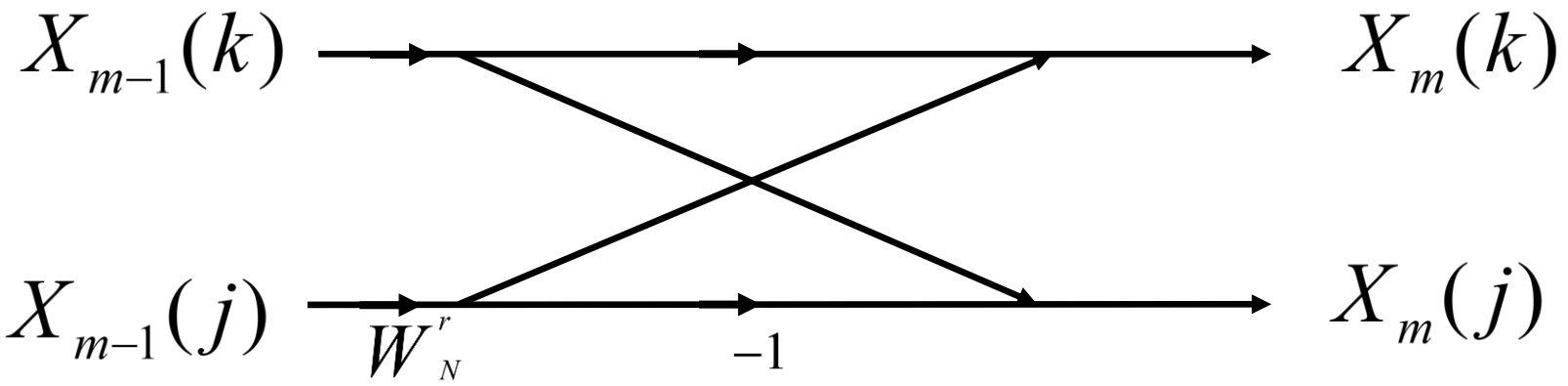
N=8按时间抽取FFT运算流图

1. 原位运算(同址运算)

每级(每列)计算都由N/2个蝶形运算构成，每个蝶形结构完成基本迭代运算：

$$X_m(k) = X_{m-1}(k) + X_{m-1}(j)W_N^r$$
$$X_m(j) = X_{m-1}(k) - X_{m-1}(j)W_N^r$$

m表示第m列迭代，k，j为数据所在行数



- 两点构成一个蝶形单元，并且这两点不再参与别的蝶形单元的运算，这种运算称为“同址运算”。
- 蝶形的两个输出值仍放回蝶形的两个输入所在的存储器中，每列的N/2个蝶形运算全部完成后，再开始下一列的蝶形运算。

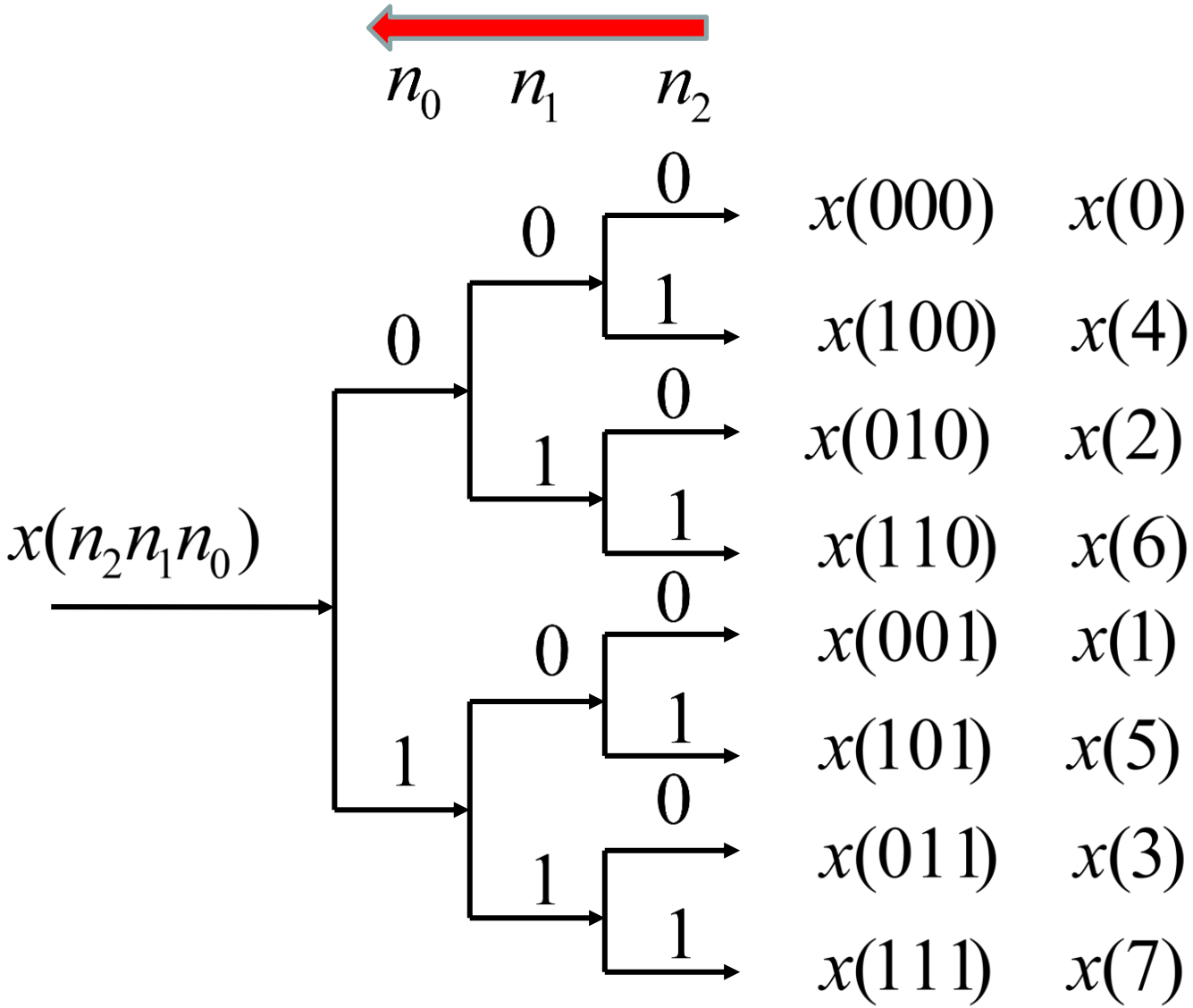
$$X_m(k) = X_{m-1}(k) + X_{m-1}(j)W_N^r$$
$$X_m(j) = X_{m-1}(k) - X_{m-1}(j)W_N^r$$

- 存储数据只需N个存储单元，原位运算结构可以节省存储单元，降低设备成本。

2. 倒位序规律

- FFT的输出X(k)按正常顺序 X(0), X(1), X(2), X(3), X(4), X(5), X(6), X(7) 排列在存储单元中，
- 输入x(n)却不是按自然顺序存储，而是按x(0), x(4), x(2), x(6), x(1), x(5), x(3), x(7)的顺序存储
- 该位序称为倒位序。

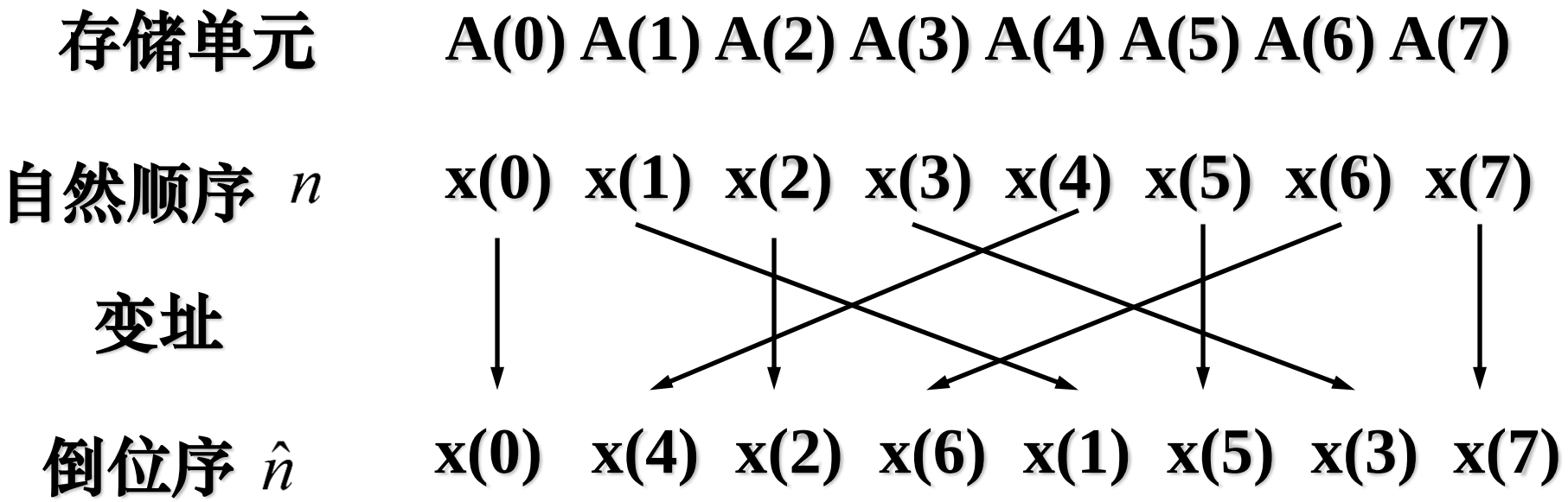
造成倒位序的原因是输入x(n)按标号n的奇偶不断抽取（分组）造成。



3. 倒位序的实现

自然顺序n	二进制数		倒位序二进制数	倒位序$\hat{n}$
0	000		000	0
1	001		100	4
2	010	码位倒置	010	2
3	011	⟺	110	6
4	100		001	1
5	101		101	5
6	110		011	3
7	111		111	7

倒位序的变址处理


存储单元
A(0) A(1) A(2) A(3) A(4) A(5) A(6) A(7)
自然顺序 n
x(0) x(1) x(2) x(3) x(4) x(5) x(6) x(7)
变址
倒位序 n̂
x(0) x(4) x(2) x(6) x(1) x(5) x(3) x(7)

处理过程：

- 当 $\hat{n}=n$ 时，$x(\hat{n})$ 和 $x(n)$ 不必交换。
- 当 $\hat{n}\neq n$ 时，$x(\hat{n})$ 和 $x(n)$ 要交换。
- 为避免交换2次，规定如下：
- 当 $\hat{n}>n$ 时，将原存放 $x(n)$ 及存放 $x(\hat{n})$ 的存储单元内的内容互换。

4. 蝶形运算两结点的“距离”

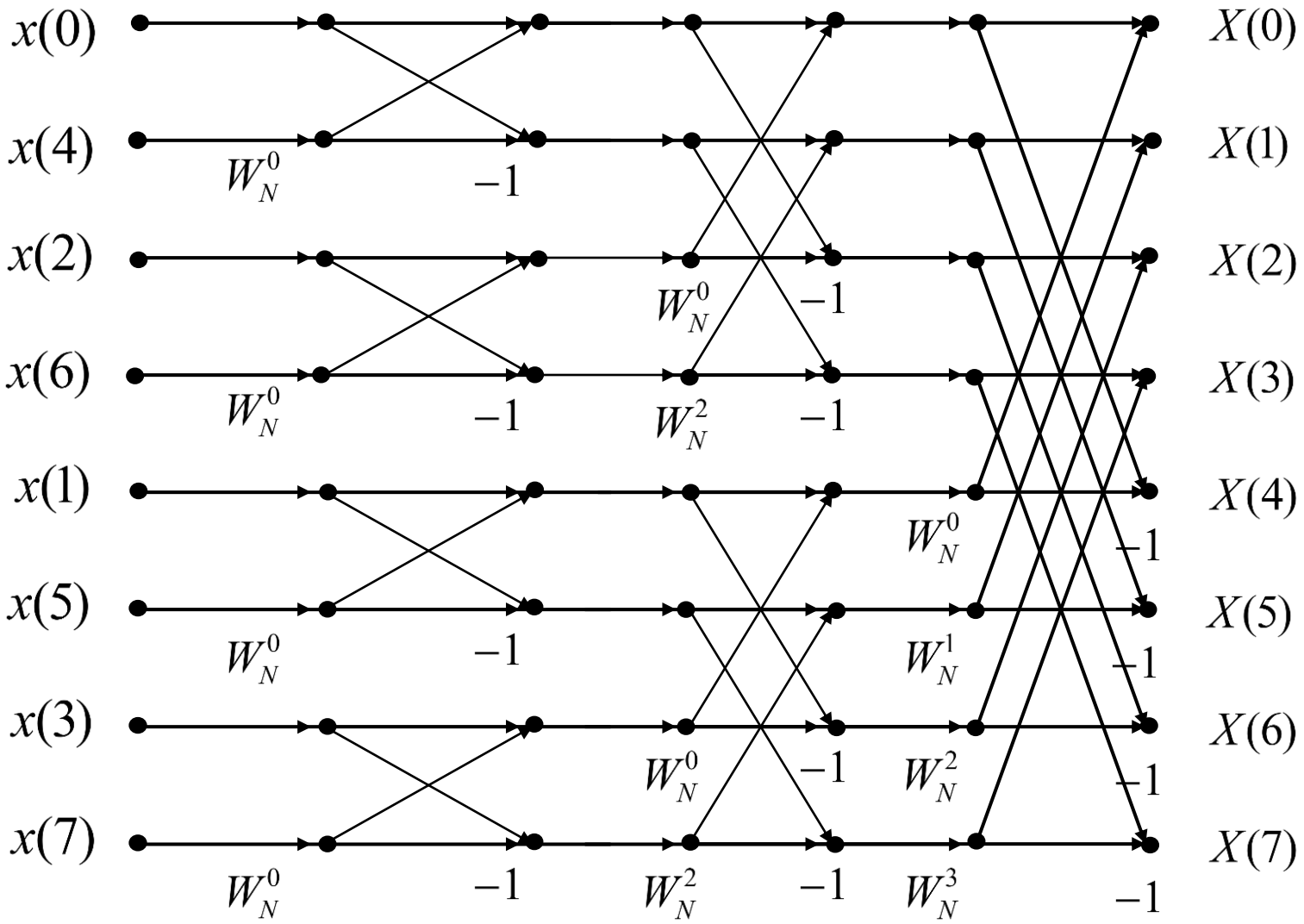
输入是倒位序，输出是自然顺序时的8点FFT，

计算第一级蝶形时(m=1)，“距离”为1 （$=2^0$）；

m=2时，“距离”为2 （$=2^1$）；

m=3时，“距离”为4 （$=2^2$）；

$N = 2^L = 2^3$, 输入是倒位序，输出是自然顺序时，可推出第m级运算，每个蝶形的两个节点“距离”为2^{m-1}。



N=8按时间抽取FFT运算流图

5. W_N^r 的确定

一个DIT蝶形运算的两节点“距离”为2^{m-1}，第m级的一个蝶形计算可表示为

$$X_m(k) = X_{m-1}(k) + X_{m-1}(k+2^{m-1})W_N^r$$

$$X_m(k+2^{m-1}) = X_{m-1}(k) - X_{m-1}(k+2^{m-1})W_N^r$$

r的求解方法：

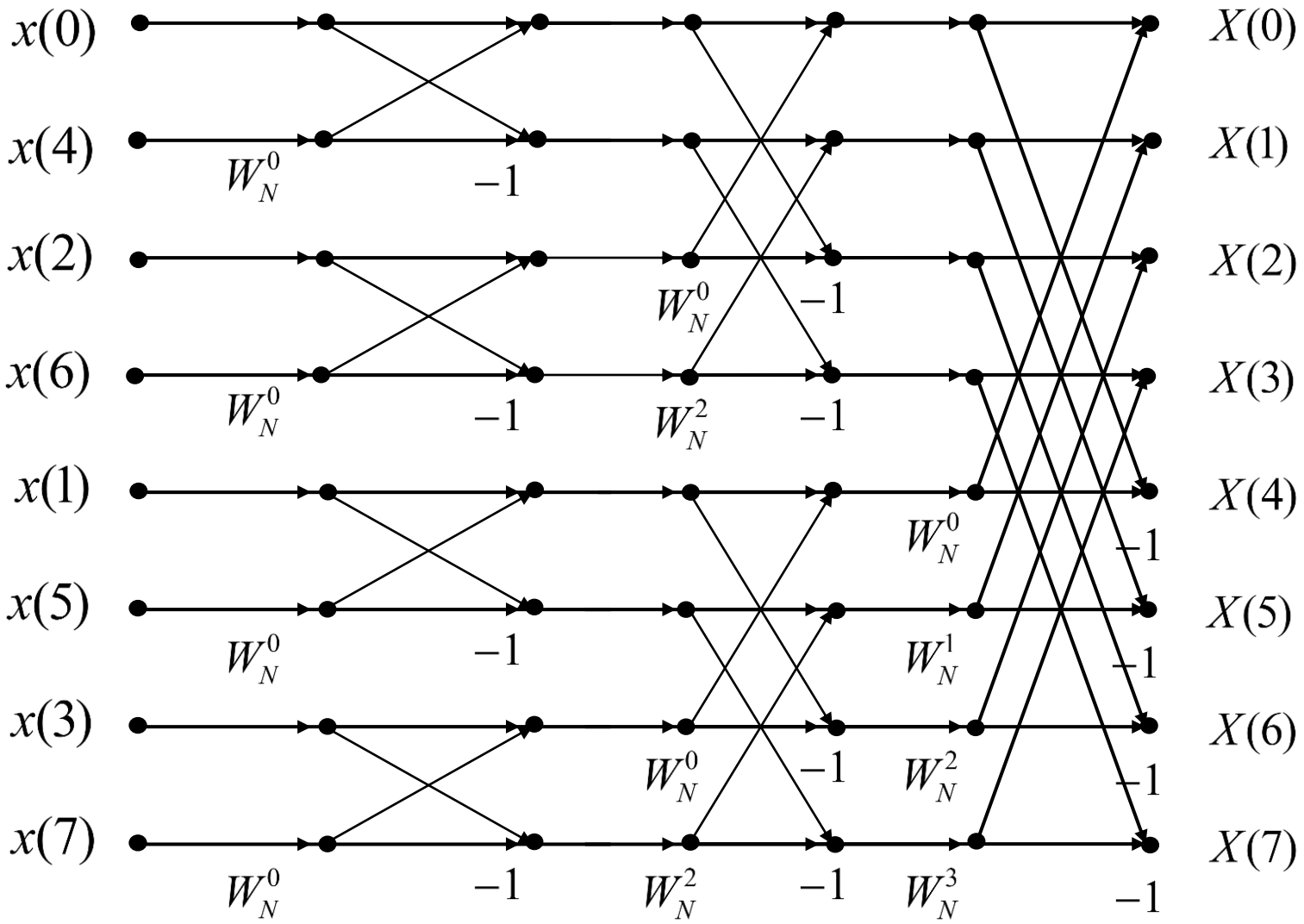
1）把蝶形运算两个节点中的第一个节点标号值k表示成L位二进制数；

2）把二进制数乘上2^{L-m}，即左移L-m位二进制数，把右边空出的补零，此数就是所求r的二进制数。

实际中，可以参考FFT流图，W_N^r 系数因子最后一列有N/2个，顺序为：

$$W_N^0, W_N^1, \mathrm{L}\ , W_N^{(\frac{N}{2}-1)}$$

其后，每向左推进一列，则取上述系数中偶数序号的那一半，一直到第一列 W_N^0 为止。



N=8按时间抽取FFT运算流图

6. 存储单元

- 输入序列x(n)需N个存储单元
- 系数 W_N^r 需N/2个存储单元
- 全部存储单元：（N+N/2）个

四、DIT的FFT算法的其他形式流图

只要保持各节点所连的支路及传输系数不变，则不论节点位置如何排列所得流图总是等效的，都是x(n) 的DFT的正确结果，只是数据的提取和存放的次序不同。

输入自然顺序，输出倒位序的流图

输入和输出皆为自然顺序的流图

参看课本：P178-179

不要求掌握，但要理解：流图不唯一！

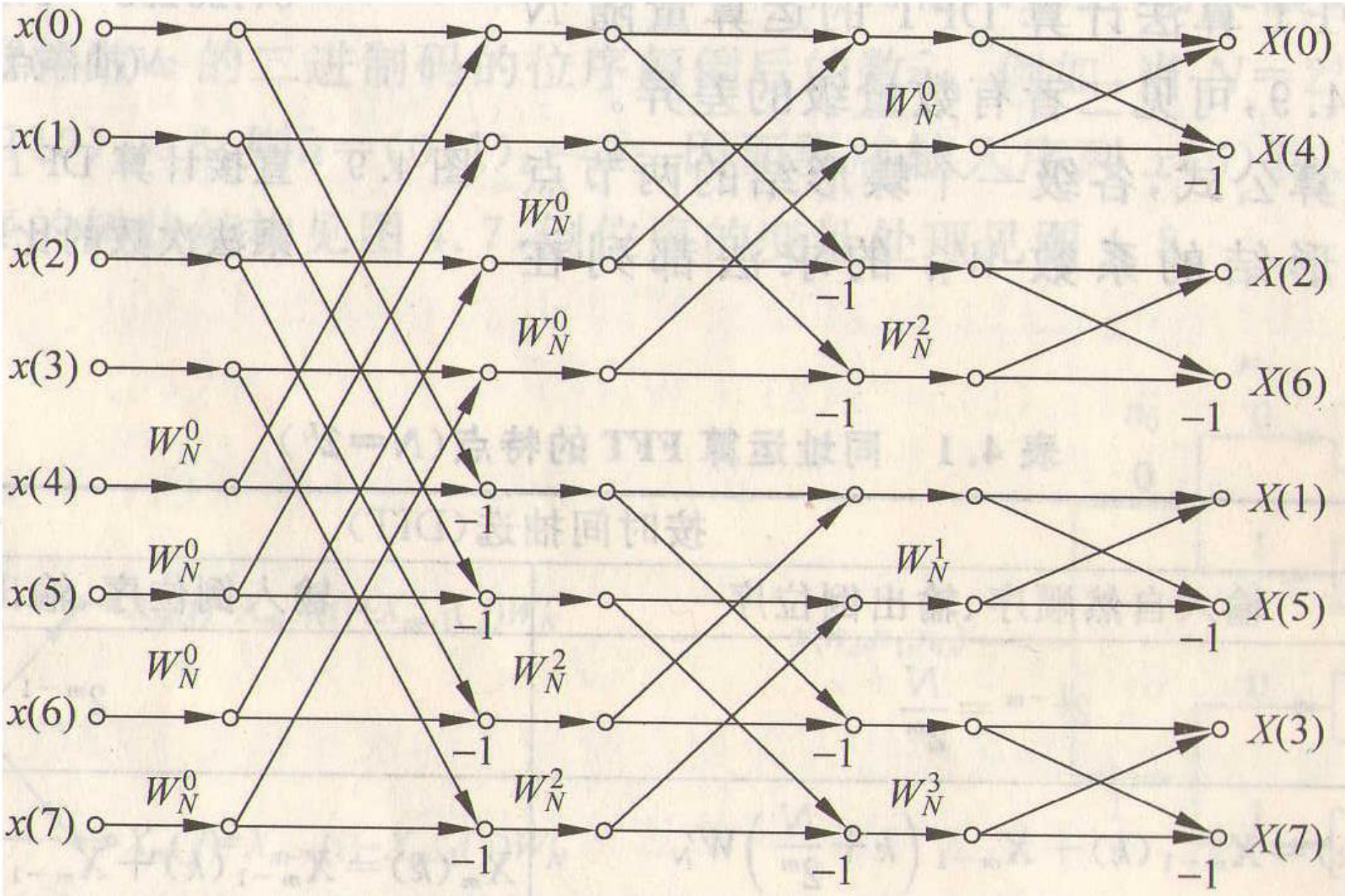


图 4.10　按时间抽选，输入自然顺序、输出倒位序的 FFT 流图

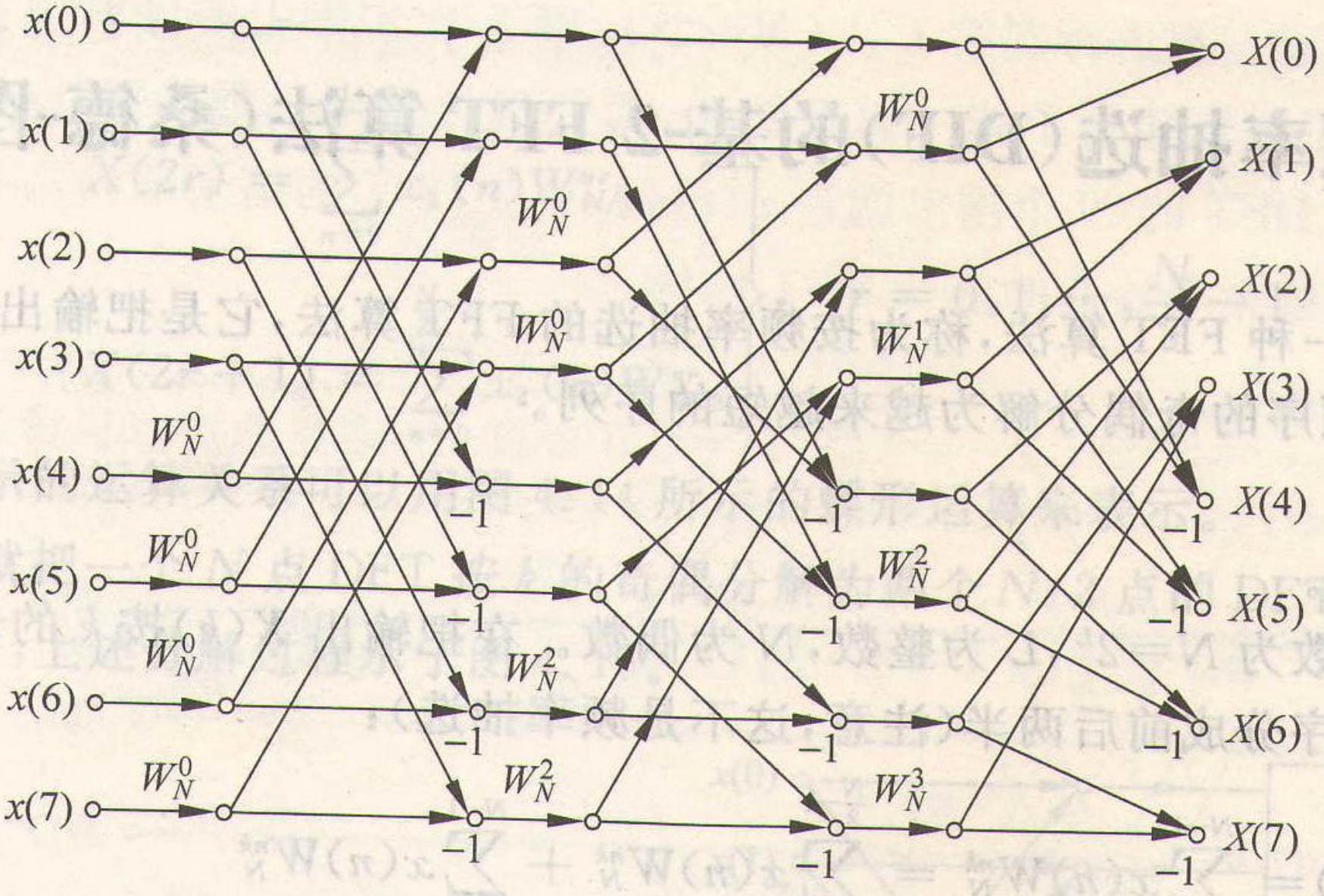


图 4.11　按时间抽选，输入输出皆为自然顺序的 FFT 流图

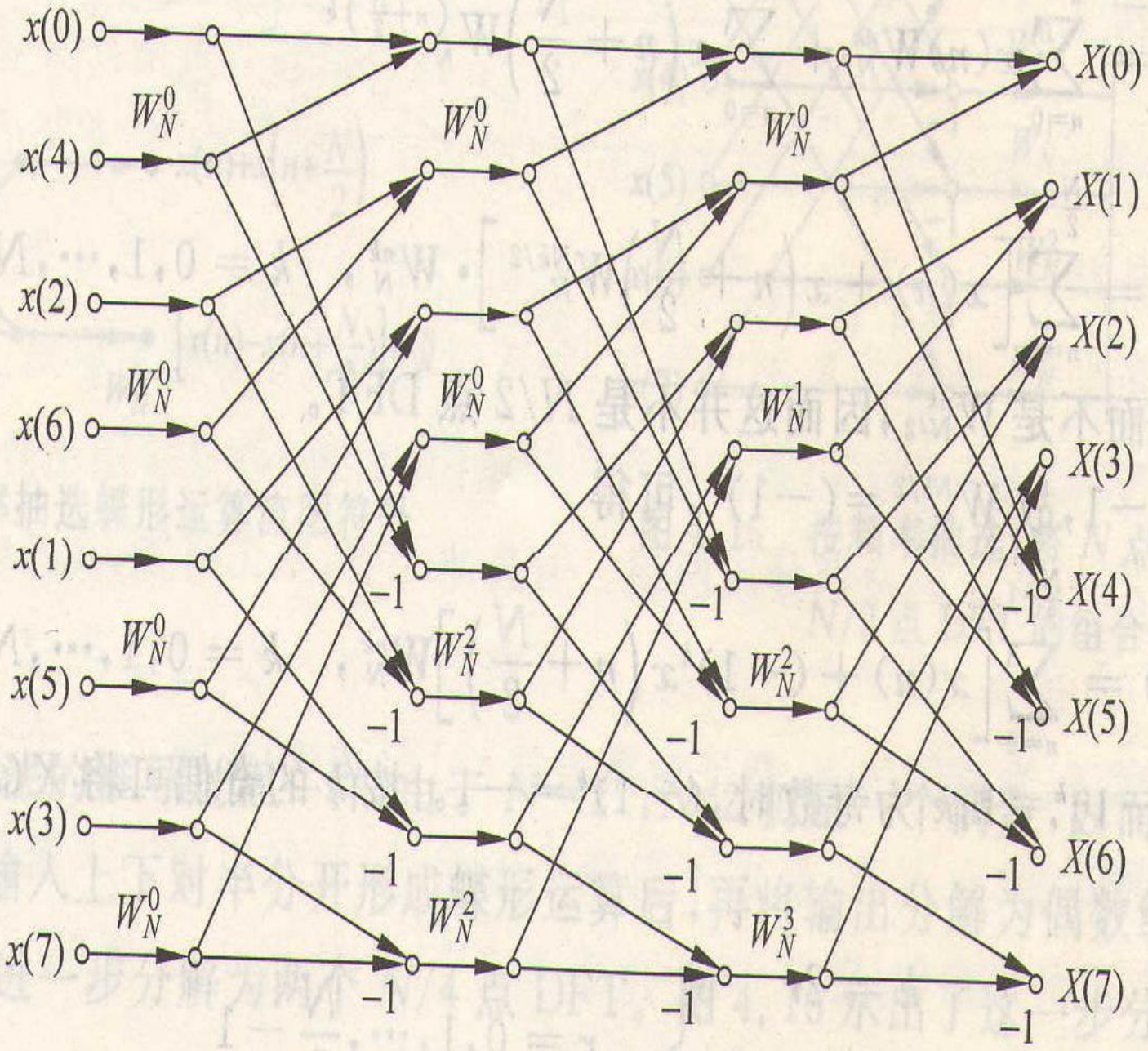


图 4.12　按时间抽选，各级具有相同几何形状，输入倒位序，输出自然顺序的 FFT 流图

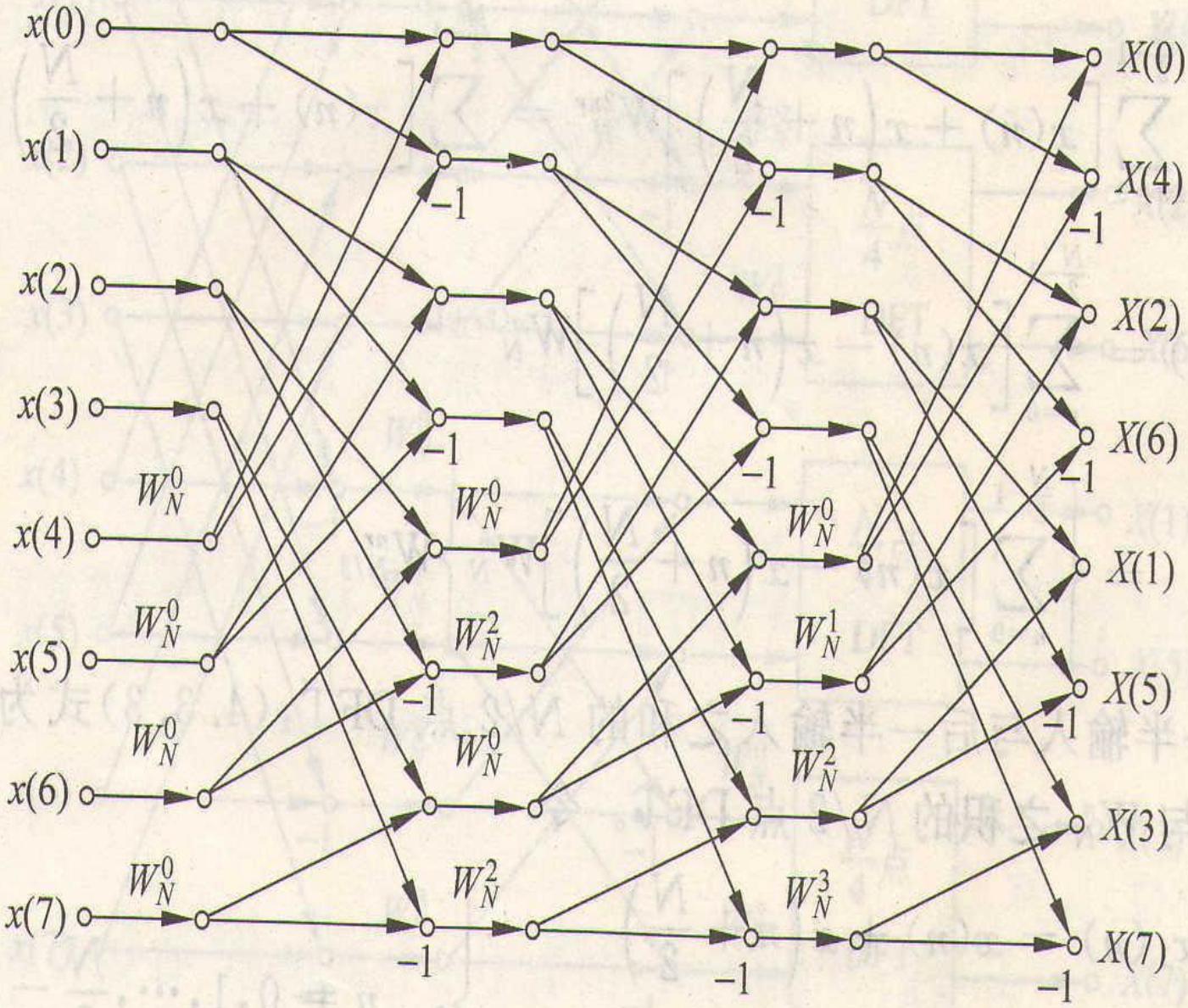


图 4.13　按时间抽选，各级具有相同几何形状，输入自然顺序、输出倒位序的 FFT 流图

4.3 频率抽取(DIF)基 2 算法

按频率抽取（DIF）的FFT算法，即把输出序列 X(k) 按照 k 的奇偶进行分解。

序列长度 $N = 2^L$, L为整数

一、算法原理

二、运算量

三、频率抽取的FFT特点

四、频率抽取的FFT算法的其它形式

一、算法原理

1. 将DFT的计算公式，不是按照 n 的奇偶进行分解，而是按照前后两部分进行分解。
2. 对DFT的计算公式，按照序号 n 的前后部分进行计算，效果就是等效于序号 k 按照奇偶进行分解。

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

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

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

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

$$k=0,1,\mathrm{L}\ ,N-1$$

上式利用了 $W_N^{N/2} = -1$

$$W_N^{Nk/2} = (-1)^k$$

k为偶数时， $(-1)^k = 1$

k为奇数时， $(-1)^k = -1$

令： $k = 2r, \quad k = 2r+1$

$$r = 0,1,\mathrm{L}\ , \frac{N}{2} - 1$$

即在频域对 k 进行奇偶分解!

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

$$X(2r)=\sum_{n=0}^{\frac{N}{2}-1}[x(n)+x(n+\tfrac{N}{2})]W_{\frac{N}{2}}^{nr} \qquad x_1(n)$$

$$X(2r+1)=\sum_{n=0}^{\frac{N}{2}-1}[x(n)-x(n+\tfrac{N}{2})]W_N^n W_{\frac{N}{2}}^{nr} \qquad x_2(n)$$

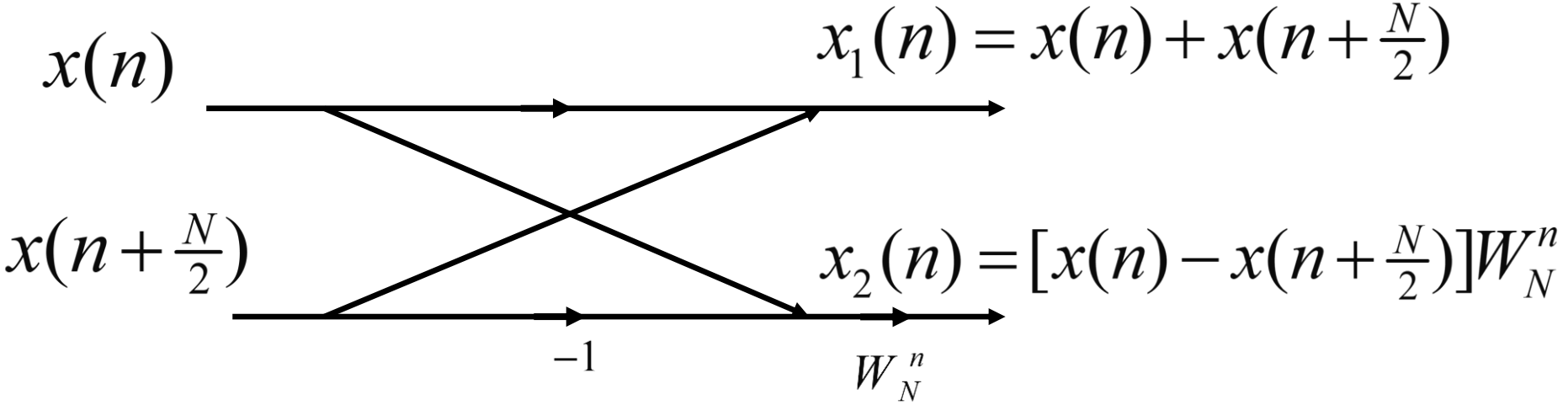
$$\left.\begin{aligned} x_1(n)&=x(n)+x(n+\tfrac{N}{2})\\ x_2(n)&=[x(n)-x(n+\tfrac{N}{2})]W_N^n \end{aligned}\right\} \quad n=0,1,\mathrm{L}\ ,\frac{N}{2}-1$$

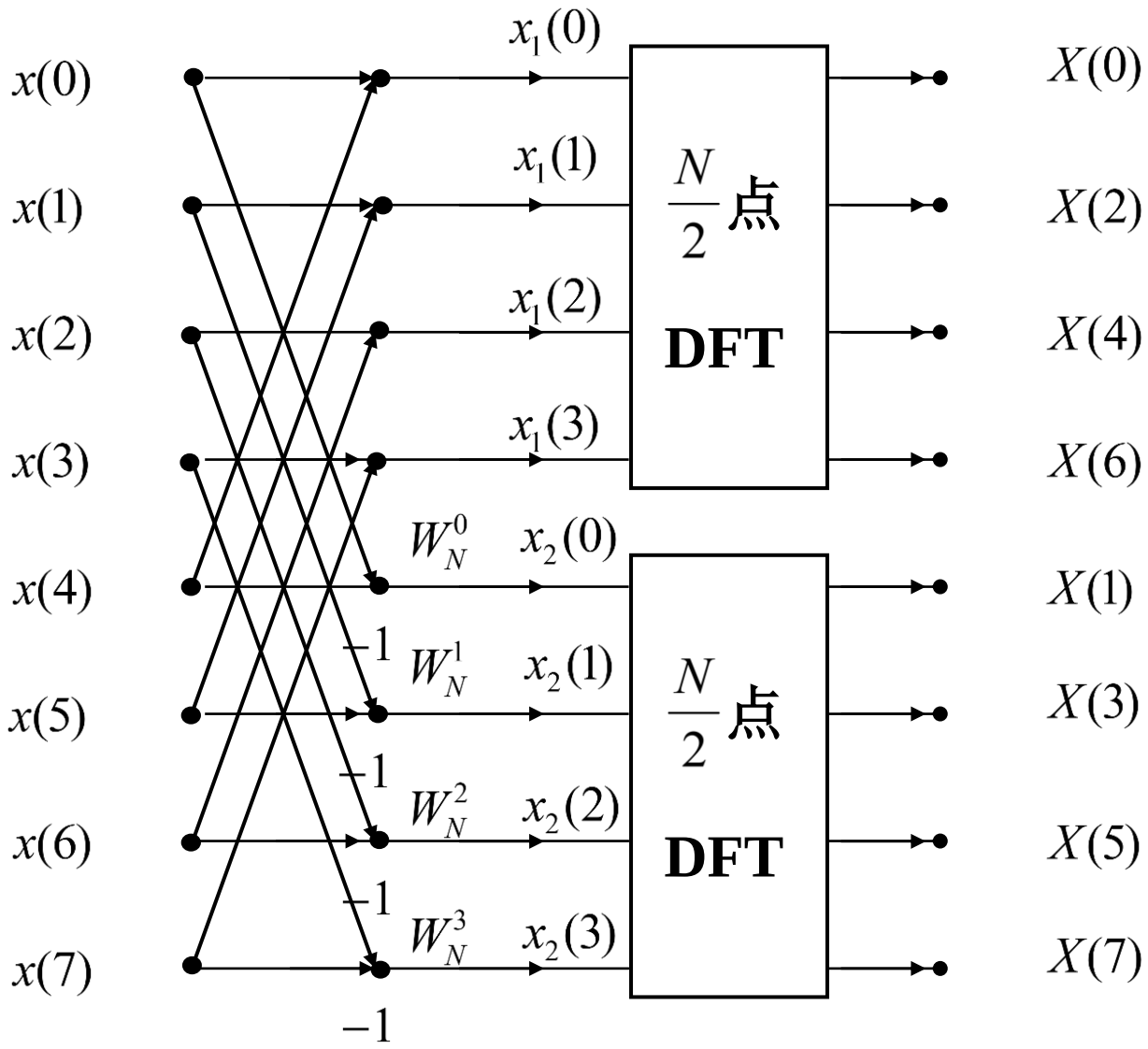
$$X(2r)=\sum_{n=0}^{\frac{N}{2}-1}x_1(n)W_{\frac{N}{2}}^{nr}$$

$$X(2r+1)=\sum_{n=0}^{\frac{N}{2}-1}x_2(n)W_{\frac{N}{2}}^{nr}$$

各是 N/2 点的 DFT

$$r=0,1,\mathrm{L}\ ,\frac{N}{2}-1$$





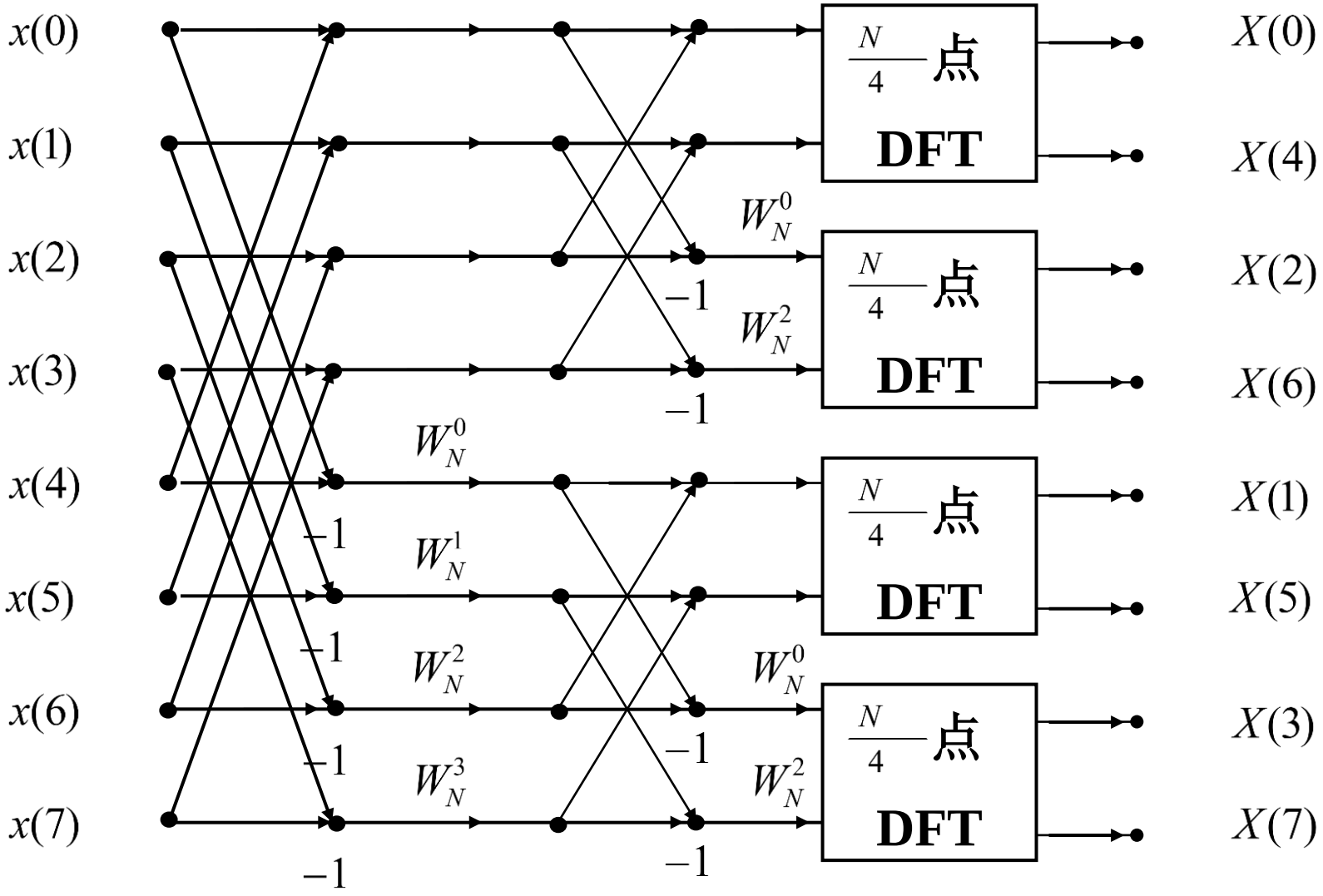
频率抽取，一个N点DFT分解为两个N/2点DFT

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

$$X(2r+1)=\sum_{n=0}^{\frac{N}{2}-1}x_2(n)W_{\frac{N}{2}}^{nr}$$

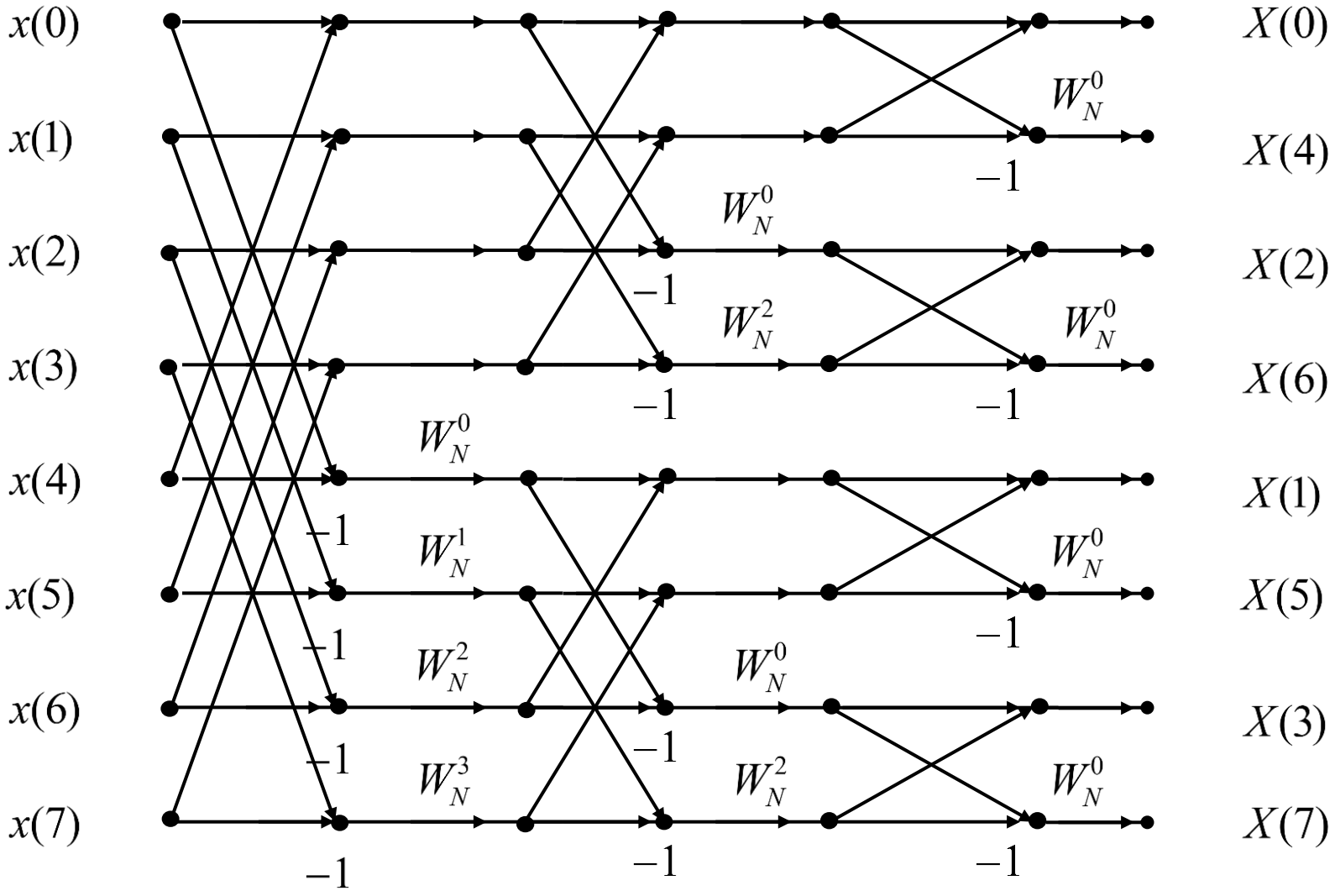
各是 N/2 点的 DFT

- **N/2仍是偶数，进一步把N/2点DFT的输出再按前后部分分解为两个N/4点的DFT；**
- **将N/2点DFT的输出上下对半分作为N/4点DFT的输入。**

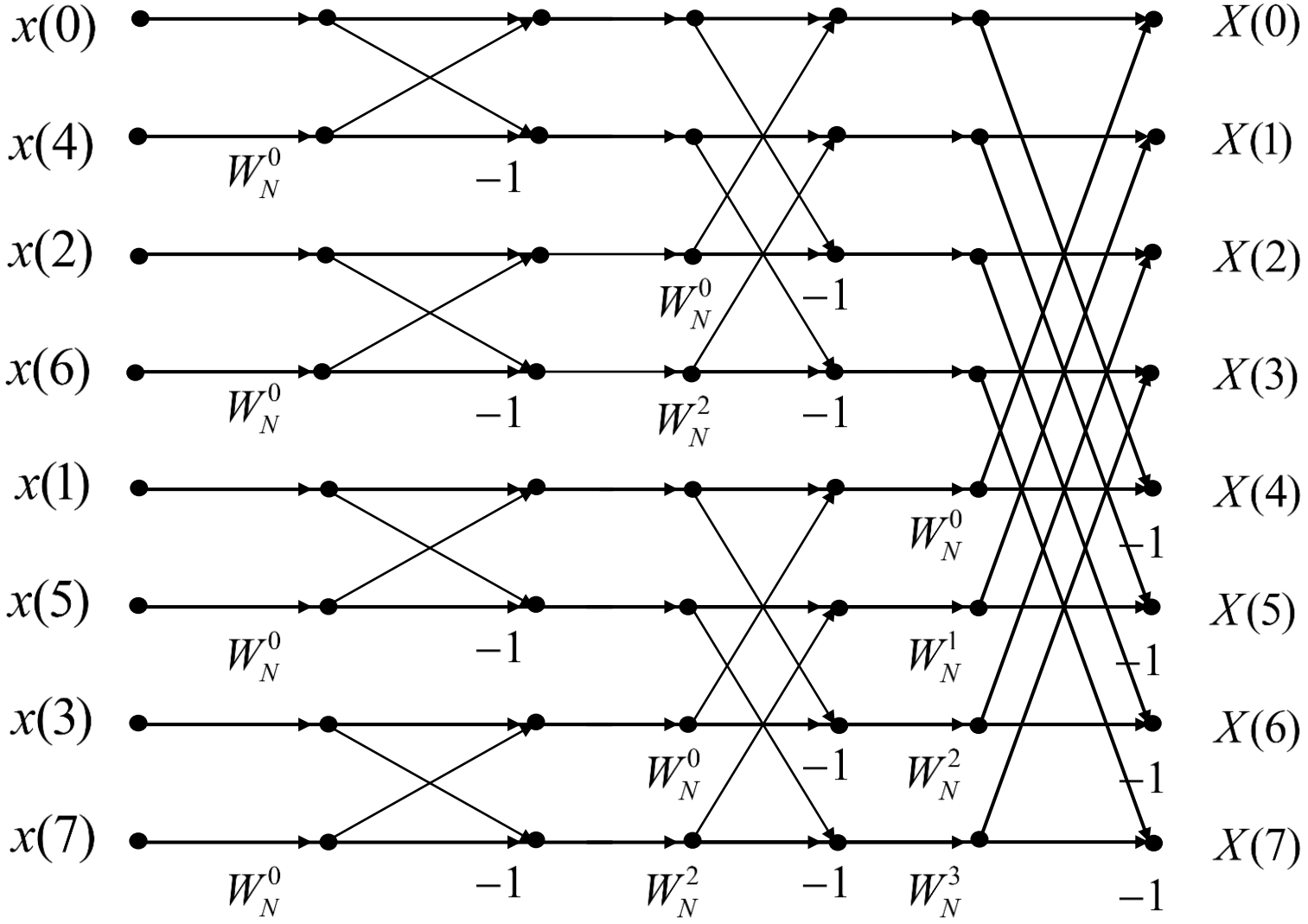


频率抽取，一个N点DFT分解为四个N/4点DFT

继续分解直到第L次，做两点DFT。这N/2个两点DFT的N个输出就是x(n)的N点DFT的结果X(k)。



频率抽取8点FFT流图



N=8按时间抽取FFT运算流图

将 n 分解： Decimation In Time, DIT 时间抽取

将 k 分解： Decimation In Freq. , DIF 频率抽取

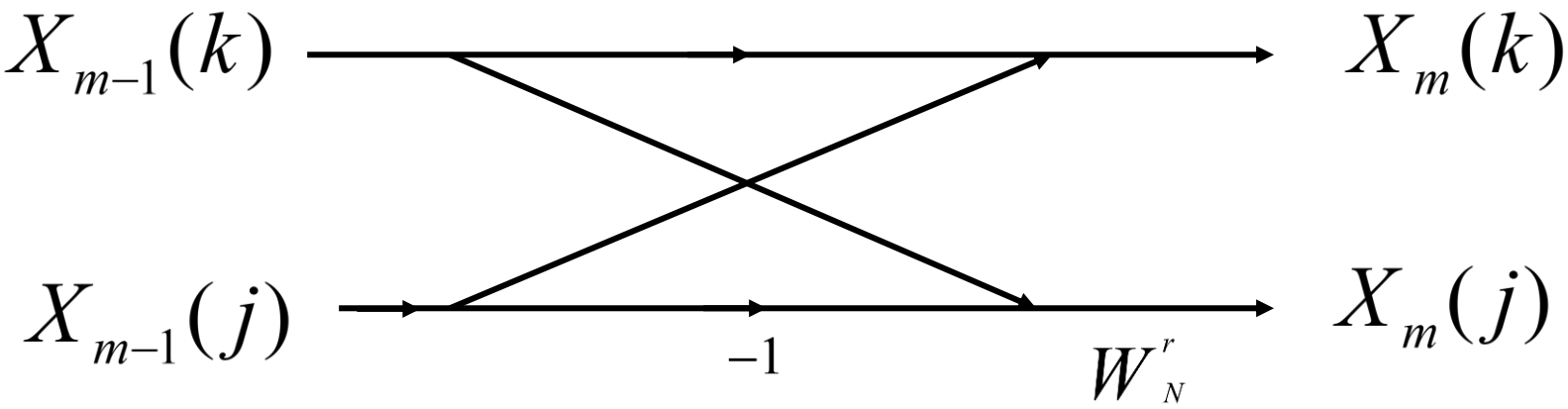
注意 DIT 和 DIF 的对偶性质。

二、原位运算

每级计算是由N/2个蝶形运算构成，每个蝶形结构完成下列基本迭代运算：

$$X_m(k) = X_{m-1}(k) + X_{m-1}(j)$$

$$X_m(j) = [X_{m-1}(k) - X_{m-1}(j)]W_N^r$$



- 每一个蝶形运算量：
  - 复数乘法： 1次
  - 复数加法：2次

- L级蝶形运算，每一级N/2个蝶形运算，故总运算量为：
  - 复数乘法：$\frac{N}{2}\log_2 N$
  - 复数加法：$N\log_2 N$

- 与DIT法计算量完全相同。

三、蝶形运算两节点间的"距离"

当 $N=2^L=2^3$ 计算

第一级蝶形时(m=1)，“距离”为 $4=\frac{N}{2^1}$ ；

m=2时，“距离”为 $2=\frac{N}{2^2}$ ；

m=3时，“距离”为 $1=\frac{N}{2^3}$ ；

可推出蝶形的两个节点“距离”为 $2^{L-m}=\frac{N}{2^m}$

四、W_N^r的计算

一个DIF蝶形运算的两节点“距离”为2^{L-m}，第m级的一个蝶形计算可表示为

$$X_m(k) = X_{m-1}(k) + X_{m-1}(k+\frac{N}{2^m})$$

$$X_m(k+\frac{N}{2^m}) = [X_{m-1}(k) - X_{m-1}(k+\frac{N}{2^m})]W_N^r$$

r的求解方法：

1）把蝶形运算两个节点中的第一个节点标号值 k 表示成二进制数；

2）把二进制数乘上 2^{m-1}，即左移 m-1 位二进制数，把右边空出的补零，此数就是所求 r 的二进制数。

实际中，可以参考DIF - FFT流图，W_N^r 系数因子第一列有N/2个，顺序为：

$$W_N^0, W_N^1, \mathrm{L}\ , W_N^{(\frac{N}{2}-1)},$$

其后，每向后推进一列，则用上述系数中偶数序号的那一半，一直到第一列 W_N^0 为止。

五、DIF与DIT的异同

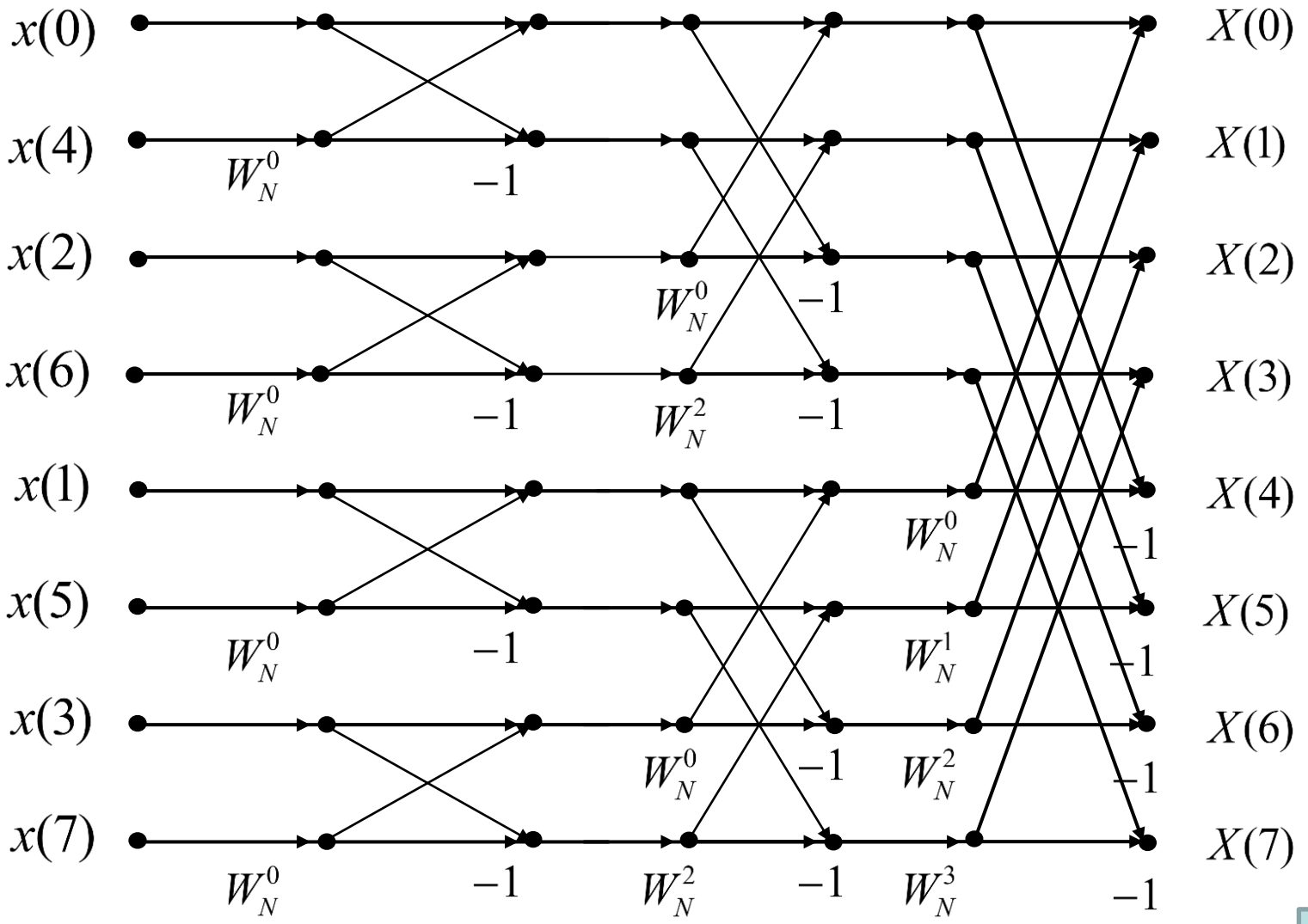
- 比较DIT和DIF流图，

（1）DIT，输出顺位序（自然顺序），输入倒位序

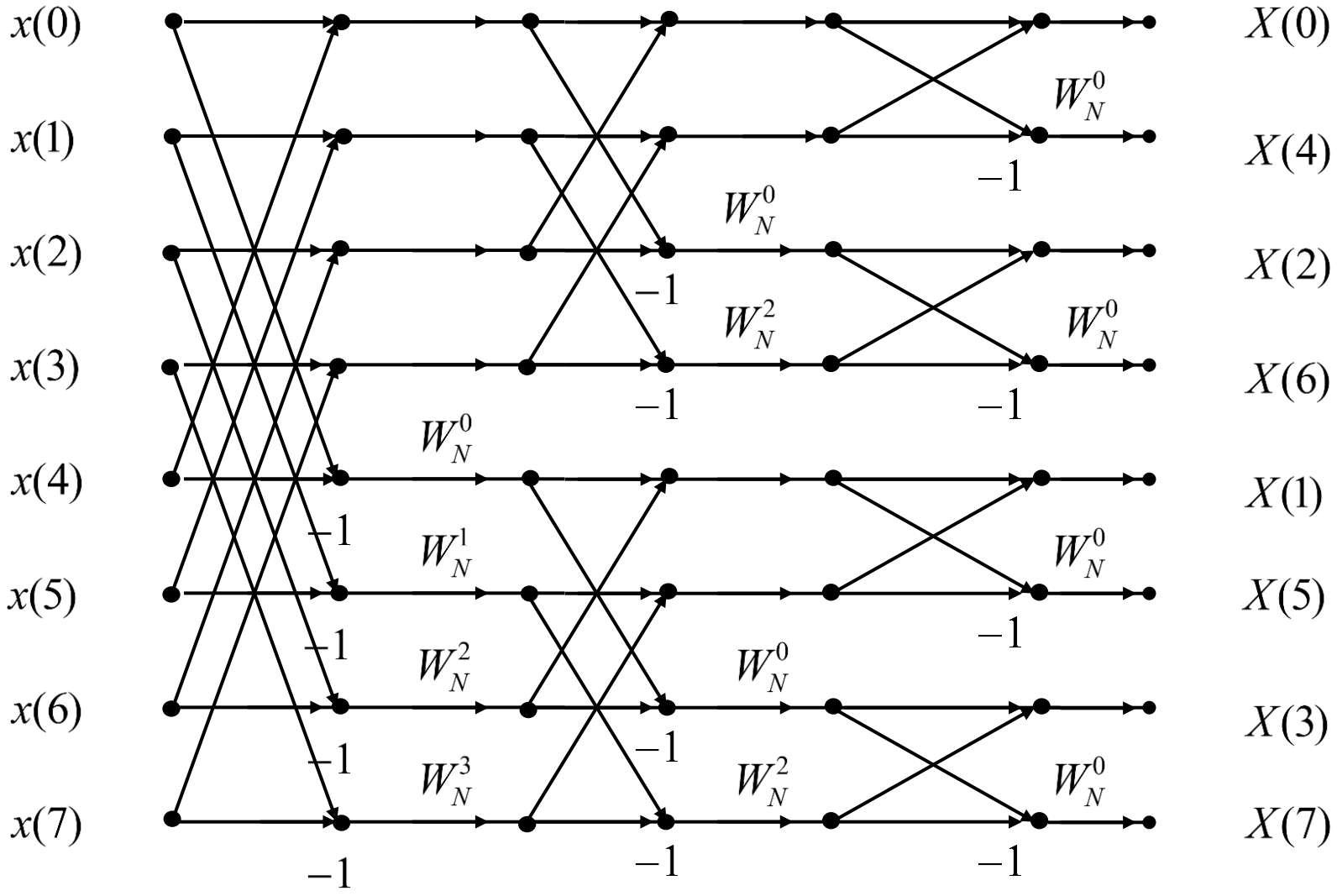
（2）DIF，输出倒位序，输入顺位序。

- 但这不是本质的区别。

因为，DIT，DIF均可以将输入、输出重新排列，得到所需的输入输出位序！（P178-179）



N=8按时间抽取FFT运算流图



频率抽取8点FFT流图

本质区别：

（1）DIF的基本蝶形与DIT的基本蝶形不同。

DIF在先作减法后作复数乘法，

DIT先作复乘后作加减法。

相同点：

1）DIF与DIT的运算量相同，有L级运算，每级运算需N/2个蝶形运算完成，每一个蝶形运算有 1 个复数乘法和 2 个复数加法，总共需要 $\frac{N}{2}\log_2 N$ 次复乘，$N\log_2 N$ 次复加 。

2）都可以进行原位运算。

- 两者的关系：
  - DIT法与DIF法的基本蝶形互为转置。
  - 转置是将流图的所有支路方向反向，交换输入输出，节点变量值不交换。

DIT和DIF流图均可以有多种形式，

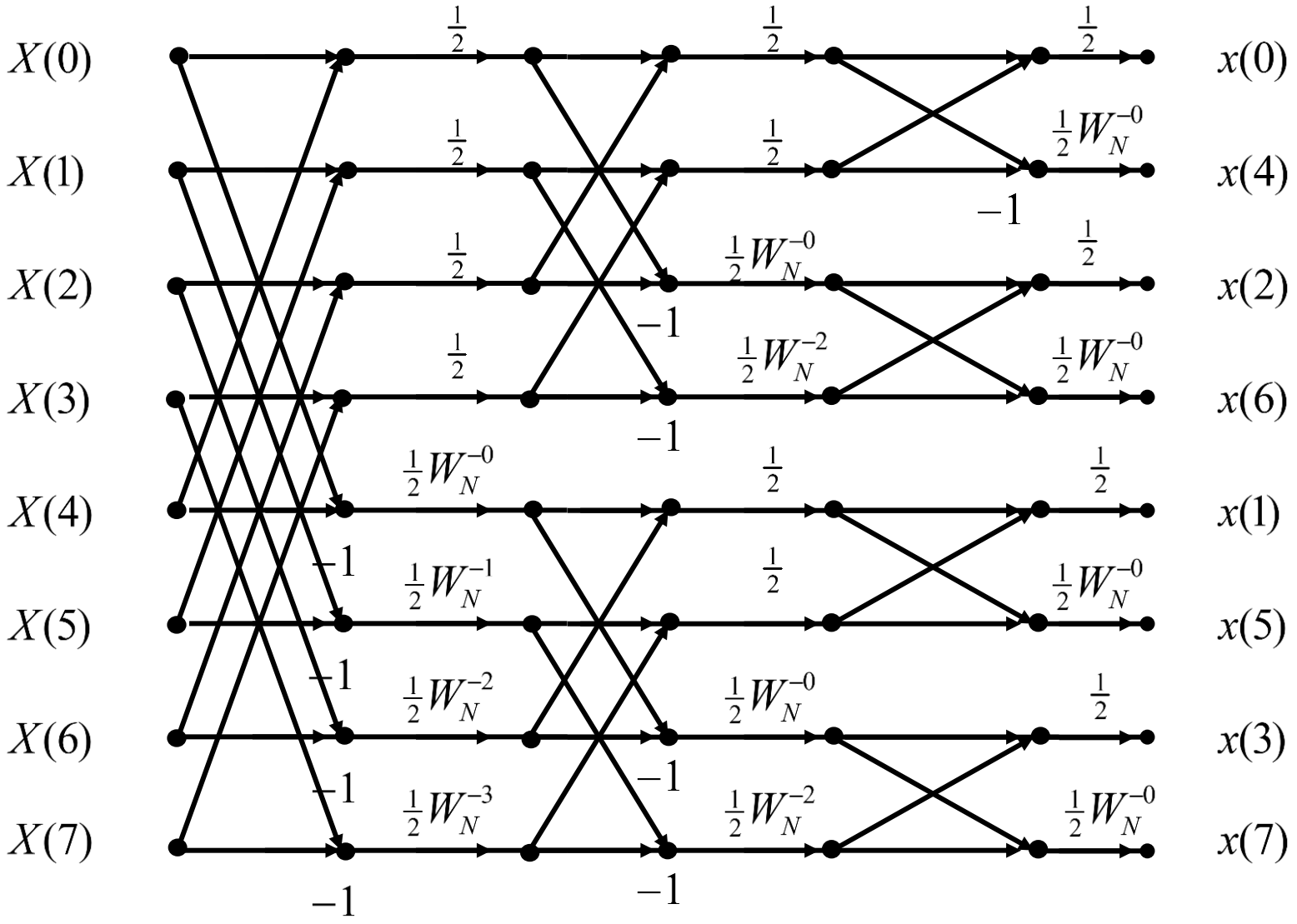
只要求基本（标准）的基2 DIT和DIF流图（图4-5和图4-17）

4.4 IDFT的快速计算方法

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

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

把DFT运算中的每一个系数W_N^{nk} 换成 W_N^{-nk}，然后再乘以1/N，则前边DIT或DIF的FFT都可以被利用来运算IDFT。



频率抽取8点IFFT流图

完全不用改变FFT的程序计算IFFT的方法：

IDFT为

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

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

$$= \frac{1}{N} DFT[X^*(k)]$$

$$x(n) = \frac{1}{N}\left\{DFT[X^*(k)]\right\}^*$$

先将X(k)取共轭，直接利用FFT计算程序，最后再将运算结果取一次共轭，并乘以1/N，就可得x(n)。

- **N为复合数的FFT算法 - 混合基FFT算法**
- **线性调频z变换算法**
- **基-4 FFT算法**

不讲!

4.5 线性卷积的FFT算法

- 线性移不变系统，输入x(n)为L点，单位抽样响应h(n)为M点，输出y(n)为

$$y(n)=\sum_{m=0}^{M-1}h(m)x(n-m)$$

因为输入值 x(n) 都必须和全部的h(n)值相乘

- y(n)为点数L+M-1点。
- 运算量：乘法次数m_d= LM次。

线性卷积的FFT算法：

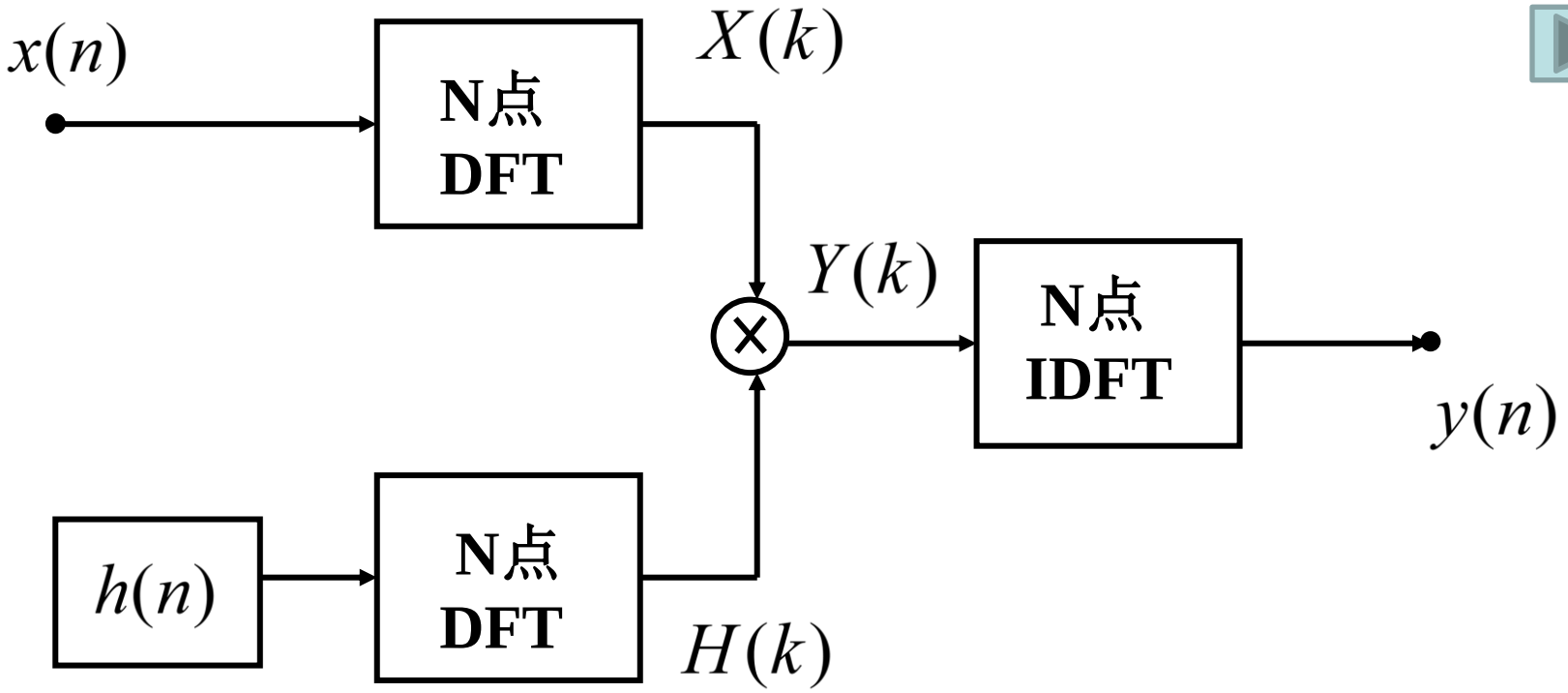
就是用圆周卷积代替线性卷积；

为了不产生混叠，充分条件是将x(n)，h(n)补零补到N＝M＋L－1。

然后计算x(n)，h(n)的圆周卷积，所得结果就是线性卷积的结果。

- 在实际应用中，会遇到一个长度很长的序列x(n)（系统输入序列）和一个长度较短的序列h(n)（系统取样响应）的线性卷积问题。
  ① 在输入数据 x(n)还没有完全采集之前，无法进行FFT计算，也就是说，会有很大的延时，不经济;
  ② 直接对长序列进行FFT时，FFT的优点不能体现。
- 故在实际中，我们总是对输入信号进行分组处理。

FIR滤波器的快速卷积结构



$$Y(k) = X(k) \bullet H(k)$$

$$y(n) = x(n) * h(n)$$

计算步骤：

（1）求 H(k) = DFT[h(n)]，N点FFT；

（2）求 X(k) = DFT[x(n)]，N点FFT；

（3）计算 Y(k) = X(k) H(k)；

（4）求 y(n) = IDFT[Y(k)]，N点FFT。

步骤(1)，(2)，(4)可以利用FFT完成。

运算量：

乘法次数

$$m_F = \frac{3}{2} N \log_2 N + N = N(1 + \frac{3}{2} \log_2 N)$$

比较直接计算线性卷积和FFT算法计算线性卷积

$$K_m = \frac{m_d}{m_F} = \frac{ML}{N(1+\frac{3}{2}\log_2 N)}$$

$$N = M + L - 1$$

若 $h(n)$ 是线性相位FIR数字滤波器，则线性卷积的乘法运算是原来的一半，为：

$$m_d = ML/2$$

比较直接计算线性卷积和FFT算法计算线性卷积

$$K_m = \frac{m_d}{m_F} = \frac{ML}{2N(1+\frac{3}{2}\log_2 N)}$$

$$N = M + L - 1$$

（1）x(n)与h(n)点数差不多，设 $L = M$ **，则** $N = 2M - 1 \approx 2M$ **，则**

$$K_m = \frac{M \times M}{2 \times 2M \times \left(1+\frac{3}{2}\log_2 2M\right)} = \frac{M}{10 + 6\log_2 M}$$

M=L	8	32	64	128	256	512	1024	2048	4096
K_m	0.29	0.80	1.39	2.46	4.41	8.01	14.62	26.95	49.95

M＝8，16，32时，圆周卷积运算量大于线性卷积；

M＝64时，两者相当；

M＝512时，圆周卷积运算速度可以快8倍；

M＝4096时，圆周卷积快50倍。

由此可见，M=L且 M 超过64时，M越长圆周卷积的优势越明显。

因此圆周卷积也称为快速卷积。

（2）x(n)的点数很多时，即L >> M时，

则N＝L+M-1≈L，这时

$$K_m = \frac{ML}{2N\left(1+\frac{3}{2}\log_2 N\right)} \approx \frac{M}{2+3\log_2 L}$$

L太大，会使比值下降，圆周卷积的优点体现不出来，

例：当L=1024, M=32时，Km=1

当L＝1024, M=16时，Km＝0.5

需采用分段卷积或分段过滤的方法。

讨论：一个短的有限长序列与一个长序列的卷积。

有两种分段卷积法（重叠相加法和重叠保留法）。

1 重叠相加法

设h(n)的点数为M，信号x(n)为很长的序列。将x(n)分解为很多段，每段为L点，L选择成和M的数量级相同，用$x_i(n)$表示x(n)的第i段：

$$x_i(n)=\begin{cases}x(n), iL\le n\le (i+1)L\\ 0 \quad \text{其他n}\end{cases} \qquad i=0,1,\mathrm{L}$$

则输入序列表示成

$$x(n)=\sum_{i=0}^{\infty}x_i(n)$$

x(n)与h(n)的线性卷积等于各$x_i(n)$与h(n)的线性卷积之和，即

$$y(n) = x(n) * h(n) = \sum_{i=0}^{\infty} x_i(n) * h(n)$$

每一个$x_i(n)$*h(n)都可以用快速卷积法计算。$x_i(n)$*h(n)为L+M-1点，故先对$x_i(n)$及h(n)补零补到N点。为了利用基-2 FFT算法，一般$N=2^m \geq L+M-1$，然后作N点圆周卷积。

由于$x_i(n)$为L点，$y_i(n)$为L+M-1点，故两段输出序列必然有M-1点发生重叠，

即前一段的后M-1点与后一段的前M-1点重叠。

因此，重叠部分相加再与不重叠的部分共同组成输出y(n)。

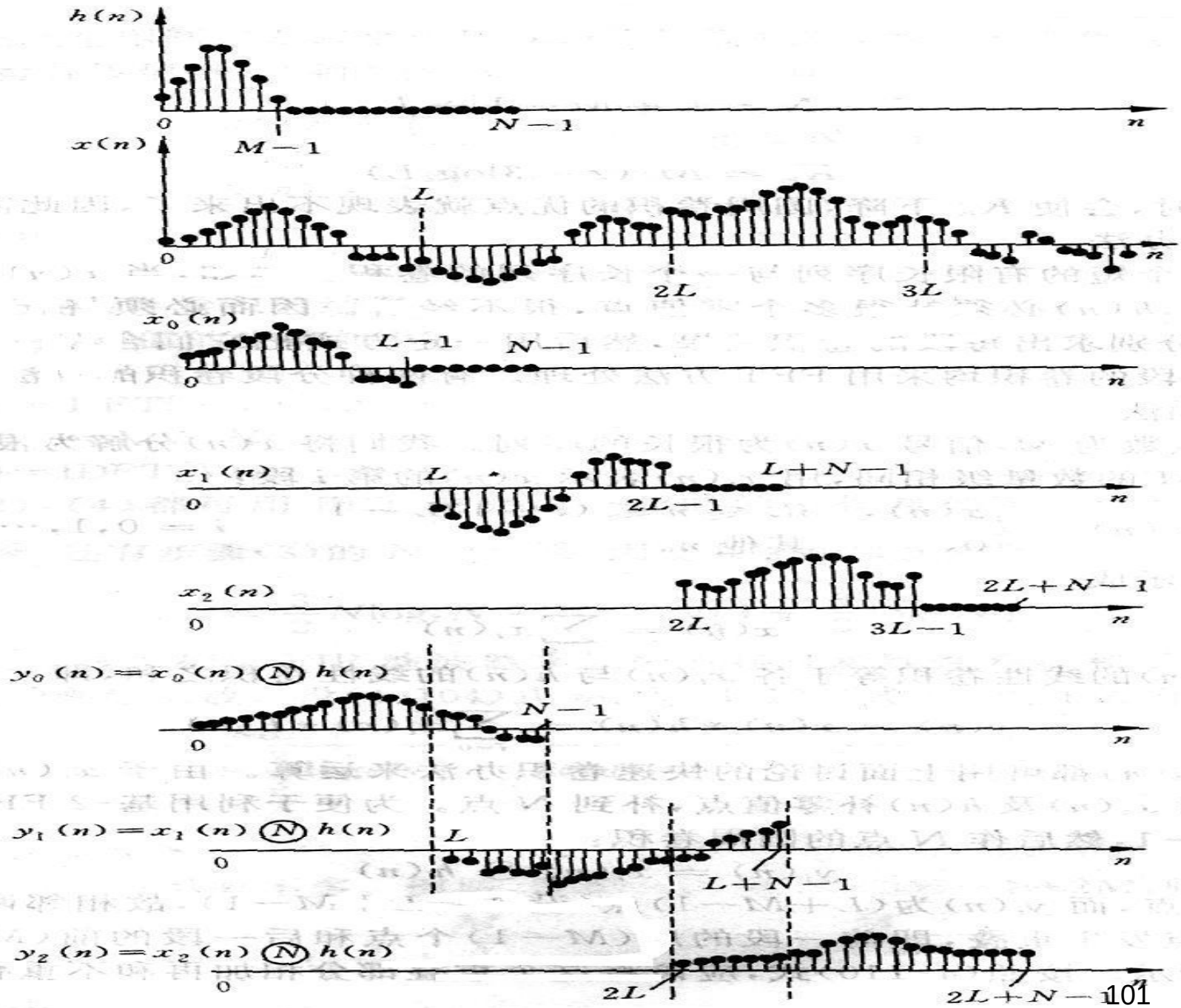


图 4-29　重叠相加法图形

FFT法实现重叠相加法的步骤：

（1）计算N点FFT，$H(k)=DFT[h(n)]$；

（2）计算N点FFT，$X_i(k)=DFT[x_i(n)]$；

（3）相乘，$Y_i(k)=X_i(k)H(k)$；

（4）计算N点IFFT，$y_i(n)=IDFT[Y_i(k)]$；

（5）将各段$y_i(n)$（包括重叠部分）相加

$$y(n)=\sum_{i=0}^{\infty}y_i(n)$$

2 重叠保留法(不要求)