100052205

# 数字信号处理

# Digital Signal Processing

李慧琦 教授

信息与电子学院

北京理工大学

Tel: +86 (10) 68918239

Email: huiqili@bit.edu.cn

# 第四章 快速傅里叶变换（FFT）

## 本章主要内容

- 快速计算DFT的基本思路
- 基2按时间抽取FFT算法
- 基2按频率抽取FFT算法
- N为复合数的FFT方法
- 分裂基FFT算法
- Chirp-Z 变换
- FFT的应用：实序列FFT算法、卷积、相关计算

## §4-1 引言

**DFT** 计算量很大, 难以应用。

**DFT**的广泛应用： **FFT +**计算机技术。

**FFT** 历史： **1965**年 Cooley-Tukey 算法

**FFT**： **Fast Fourier Transform**

各种计算**DFT**的快速算法

## §4-2 直接计算DFT的问题和改善DFT运算效率的途径

### 一、直接计算DFT的问题

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

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

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

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

**DFT 计算量**

复数 *： $N^2$

复数 +： $N(N-1)$

设 $N=1024 \qquad 8092$

$=10^6 \qquad 65.5\times10^6$

$=10^6 \qquad 65.5\times10^6$

$X(k)=X_R(k)+jX_I(k)$

$$W_N^{kn}=\cos(2\pi kn/N)-j\sin(2\pi kn/N)$$

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

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

三角函数： $2N^2$

实数 *： $4N^2$

实数 +： $N*2*(2N-1)$

Example: compute 4点序列{2, 3, 3, 2}之DFT

$$X[m]=\sum_{k=0}^{N-1}x[k]W_N^{km},\qquad m=0,1,\cdots,N-1$$

$$X[0]=2W_N^0+3W_N^0+3W_N^0+2W_N^0=10$$

$$X[1]=2W_N^0+3W_N^1+3W_N^2+2W_N^3=-1-j$$

$$X[2]=2W_N^0+3W_N^2+3W_N^4+2W_N^6=0$$

$$X[3]=2W_N^0+3W_N^3+3W_N^6+2W_N^9=-1+j$$

| N | DFT |
|---|---|
| 4 | 16 |
| 32 | 1024 |
| 128 | 16384 |
| 1024 | 1048576 |

复数加法: $N(N-1)$ 复数乘法: $N^2$

如何提高计算效率?

# 二、改善DFT运算效率的基本途径

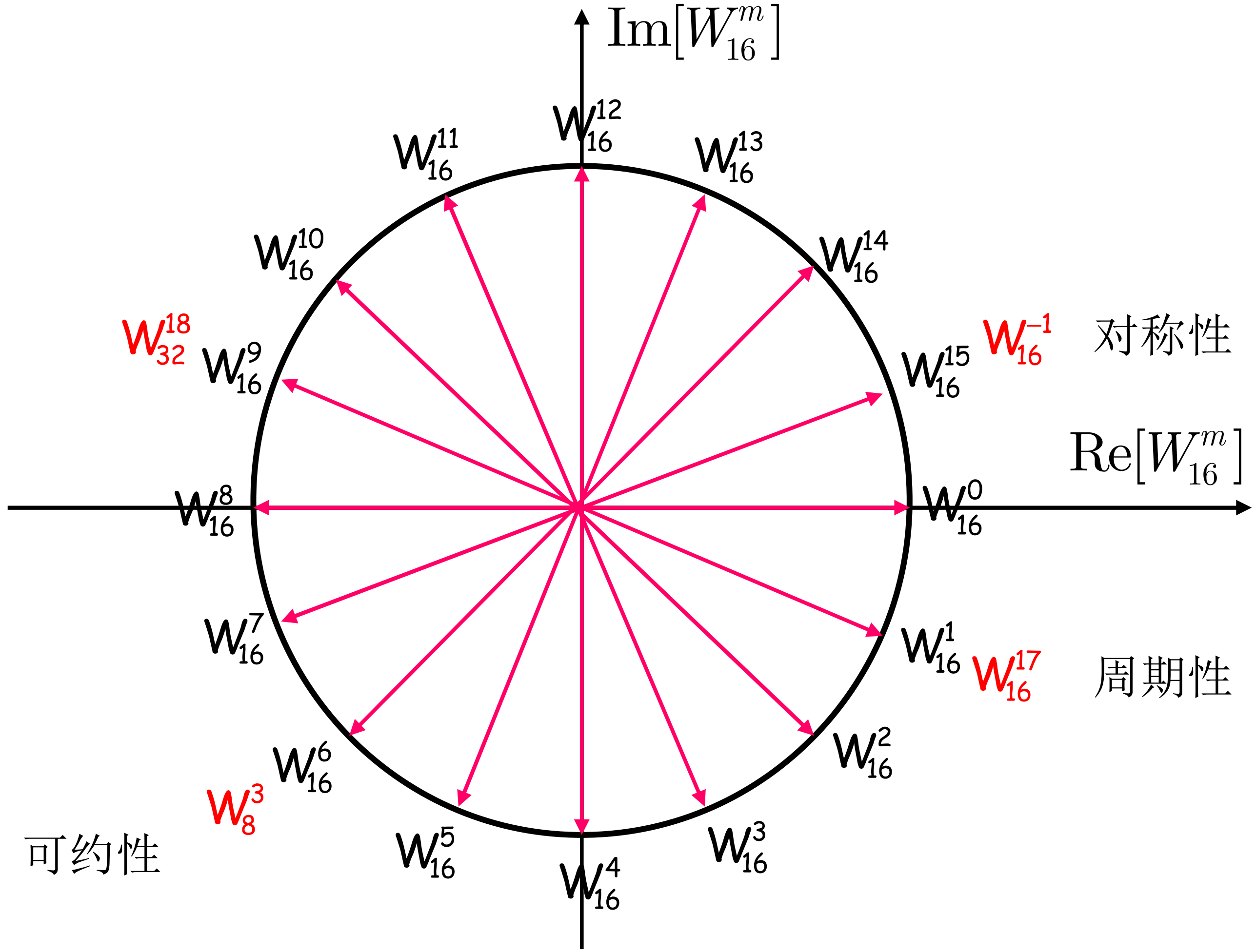
1.利用$W_N^{kn}$的特性

对称性 $(W_N^{nk})^* = W_N^{-nk} = W_N^{(N-n)k} = W_N^{n(N-k)}$

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

可约性 $W_N^{nk} = W_{mN}^{mnk}$ $\quad W_N^{nk} = W_{N/m}^{nk/m}$

特殊点 $W_N^0 = W_N^N = 1 \quad W_N^{N/2} = -1 \quad W_N^{(k+N/2)} = -W_N^k$



$\mathrm{Im}[W_{16}^{m}]$
$\mathrm{Re}[W_{16}^{m}]$
$W_{16}^{0}$
$W_{16}^{1}$
$W_{16}^{2}$
$W_{16}^{3}$
$W_{16}^{4}$
$W_{16}^{5}$
$W_{16}^{6}$
$W_{16}^{7}$
$W_{16}^{8}$
$W_{16}^{9}$
$W_{16}^{10}$
$W_{16}^{11}$
$W_{16}^{12}$
$W_{16}^{13}$
$W_{16}^{14}$
$W_{16}^{15}$
$W_{16}^{-1}$
对称性
$W_{16}^{17}$
周期性
$W_{32}^{18}$
$W_{8}^{3}$
可约性

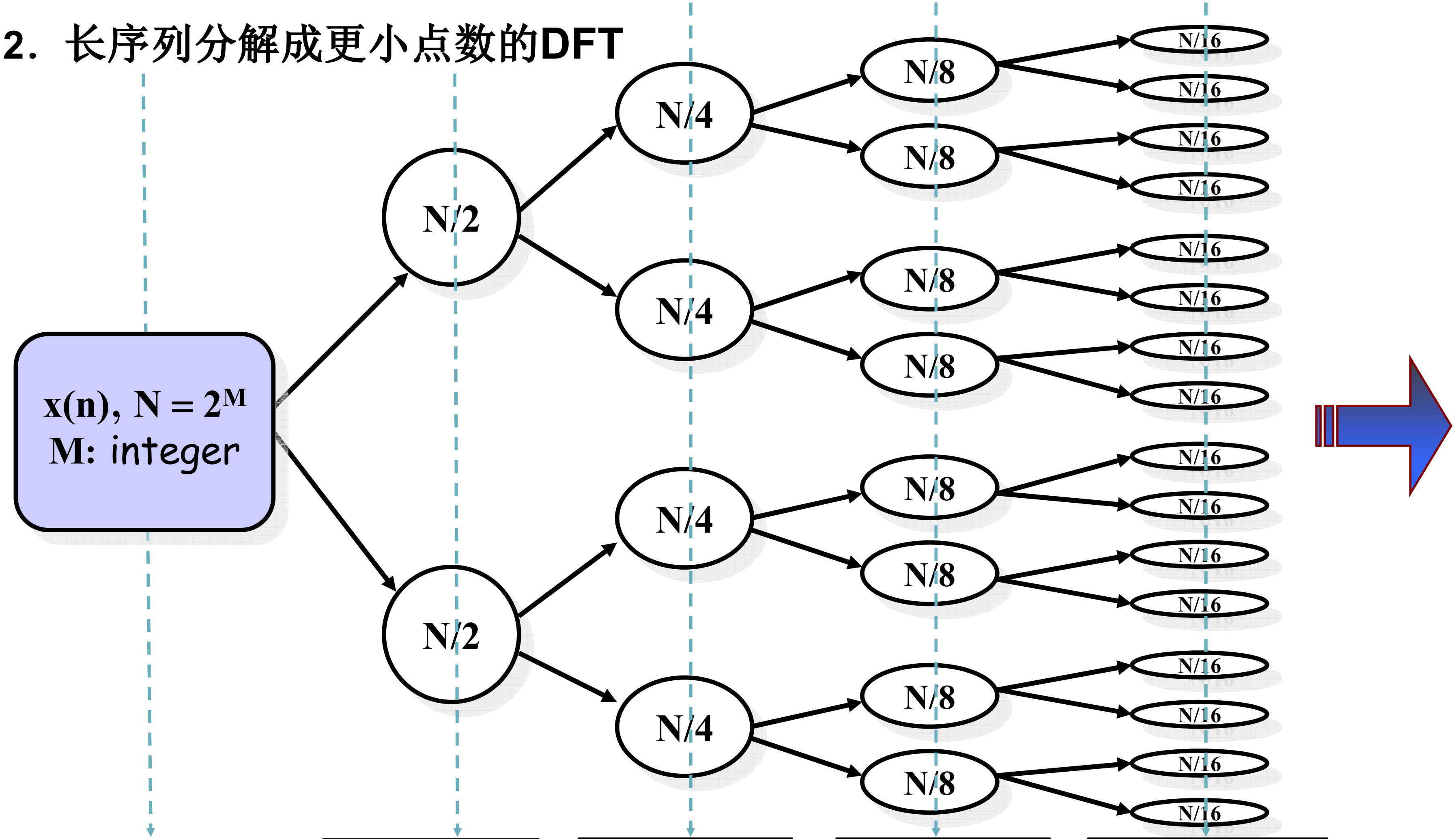
$$X[0] = 2W_4^0 + 3W_4^0 + 3W_4^0 + 2W_4^0 = 10$$

$$X[1] = 2W_4^0 + 3W_4^1 + 3W_4^2 + 2W_4^3 = -1 - j$$

$$X[2] = 2W_4^0 + 3W_4^2 + 3W_4^4 + 2W_4^6 = 0$$

$$X[3] = 2W_4^0 + 3W_4^3 + 3W_4^6 + 2W_4^9 = -1 + j$$

**2．长序列分解成更小点数的DFT**




乘法次数 $\boxed{N^2}$ $\boxed{2\left(\frac{N}{2}\right)^2=\frac{N^2}{2}}$ $\boxed{4\left(\frac{N}{4}\right)^2=\frac{N^2}{4}}$ $\boxed{8\left(\frac{N}{8}\right)^2=\frac{N^2}{8}}$ $\boxed{16\left(\frac{N}{16}\right)^2=\frac{N^2}{16}}$

**分而治之**

将序列逐次分解为一组子序列，利用旋转因子的特性，由子序列的离散傅立叶变换来实现整个序列的离散傅立叶变换

** **按时间抽取 (Decimation in time) DIT-FFT**

$$x[n] \to \begin{cases} x[2r] \\ x[2r+1] \end{cases} \qquad r = 0, 1, \cdots, \frac{N}{2} - 1$$

- Cooley-Tukey (库利-图基,美,1965 )

** **按频率抽取 (Decimation in frequency) DIF-FFT**

$$X[k] \to \begin{cases} X[2k] \\ X[2k+1] \end{cases}$$

- Sand-Tukey (桑德-图基,美,1966 )

** **分裂基方法**

- Duhamel-Hollmann (杜哈梅尔-霍尔曼,法,1984)

# §4-3 按时间抽取(DIT)的FFT算法

Jame

## An Algorithm for the Machine Calculation of Complex Fourier Series

By James W. Cooley and John W. Tukey

An efficient method for the calculation of the interactions of a $2^m$ factorial experiment was introduced by Yates and is widely known by his name. The generalization to $3^m$ was given by Box et al. [1]. Good [2] generalized these methods and gave elegant algorithms for which one class of applications is the calculation of Fourier series. In their full generality, Good's methods are applicable to certain problems in which one must multiply an $N$-vector by an $N \times N$ matrix which can be factored into $m$ sparse matrices, where $m$ is proportional to $\log N$. This results in a procedure requiring a number of operations proportional to $N \log N$ rather than $N^2$. These methods are applied here to the calculation of complex Fourier series. They are useful in situations where the number of data points is, or can be chosen to be, a highly composite number. The algorithm is here derived and presented in a rather different form. Attention is given to the choice of $N$. It is also shown how special advantage can be obtained in the use of a binary computer with $N = 2^m$ and how the entire calculation can be performed within the array of $N$ data storage locations used for the given Fourier coefficients.

Consider the problem of calculating the complex Fourier series

$$X(j) = \sum_{k=0}^{N-1} A(k) \cdot W^{jk}, \qquad j = 0, 1, \cdots, N-1, \tag{1}$$

where the given Fourier coefficients $A(k)$ are complex and $W$ is the principal $N$th root of unity,

$$W = e^{2\pi i/N}. \tag{2}$$

A straightforward calculation using (1) would require $N^2$ operations where "operation" means, as it will throughout this note, a complex multiplication followed by a complex addition.

The algorithm described here iterates on the array of given complex Fourier amplitudes and yields the result in less than $2N \log_2 N$ operations without requiring more data storage than is required for the given array $A$. To derive the algorithm, suppose $N$ is a composite, i.e., $N = r_1 \cdot r_2$. Then let the indices in (1) be expressed

$$\begin{aligned} j &= j_1 r_1 + j_0, \quad & j_0 &= 0, 1, \cdots, r_1 - 1, \quad & j_1 &= 0, 1, \cdots, r_2 - 1, \\ k &= k_1 r_2 + k_0, \quad & k_0 &= 0, 1, \cdots, r_2 - 1, \quad & k_1 &= 0, 1, \cdots, r_1 - 1. \end{aligned} \tag{3}$$

Then, one can write

$$X(j_1, j_0) = \sum_{k_0} \sum_{k_1} A(k_1, k_0) \cdot W^{jk_1 r_2} W^{jk_0}. \tag{4}$$

Received August 17, 1964. Research in part at Princeton University under the sponsorship of the Army Research Office (Durham). The authors wish to thank Richard Garwin for his essential role in communication and encouragement.



Tukey



"An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, 19 (1965), 297-301

## 一、算法原理

### 基数

$N=r_1 r_2 r_3 \ldots r_v$

如果 $r_1=r_2=r_3=\ldots=r_v=r$ $N=r^v$

r: radix of FFT （基数）

## 基-2 FFT (Radix-2 FFT)

$N = 2^{\nu}$ (若$N \neq 2^{\nu}$，可通过补零达到)

由$DFT$的定义:

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

$\frac{N}{2} - x_1(n)$ $\quad DFT$

$\boldsymbol{x(n)}$

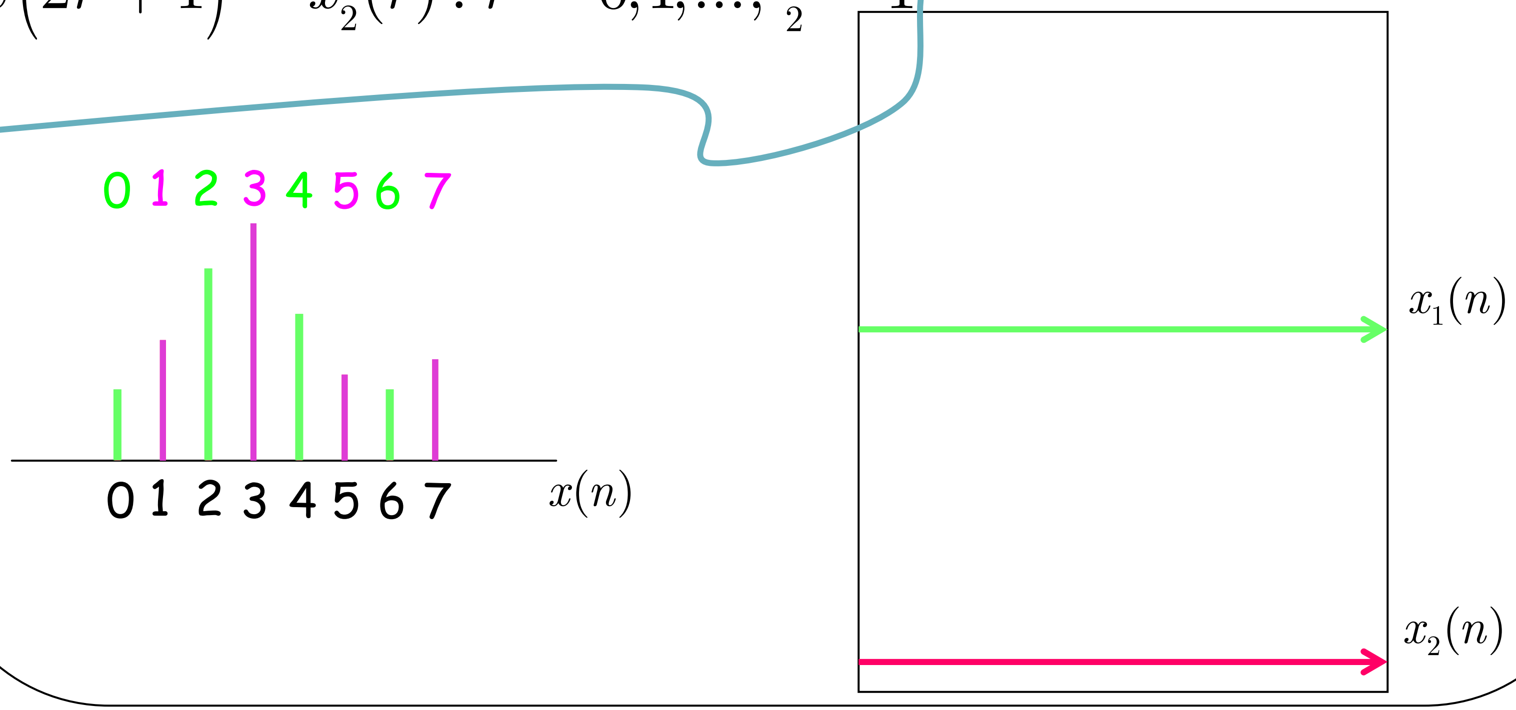
**N点*DFT*** ?

$\frac{N}{2} - x_2(n)$ $\quad DFT$

设序列点数 $N=2^v$, 为整数 (若不满足，则补零)

**将序列 $x(n)$ 按 $n$ 的奇偶分成两组：**

$$x(2r) = x_1(r)$$
$$x(2r+1) = x_2(r) : r = 0, 1, \ldots, \frac{N}{2} - 1$$



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

$x_2(n)=x(2r+1) \qquad r=0,1,\ldots,\frac{N}{2}-1$

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

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

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

$$=X_1(k)+W_N^kX_2(k)$$

式中:

$X_1(k)=DFT[x_1(n)], 0\le k\le\frac{N}{2}-1$

$X_2(k)=DFT[x_2(n)], 0\le k\le\frac{N}{2}-1$

可见：

$$\begin{tikzcd} & \frac{N}{2}-DFT \arrow[dr, dash] & & \\ N-DFT \arrow[ur] \arrow[dr] & & \cdot \arrow[r] & N-DFT \\ & \frac{N}{2}-DFT \arrow[ur, dash] & & \end{tikzcd}$$

$$\begin{tikzcd} X_1(k) \arrow[dr, dash] & & \\ & \cdot \arrow[r] & X(k), \quad 0 \le k \le \frac{N}{2}-1 \\ X_2(k) \arrow[ur, dash] & & \end{tikzcd}$$

$$0 \le k \le \frac{N}{2}-1$$

问题：$\frac{N}{2} \le k \le N-1$时，$X(k)=?$

$X_1(k)$，$X_2(k)$周期为$\dfrac{N}{2}$

$$X_1(k+\frac{N}{2}) = X_1(k)\,, \qquad 0 \le k \le \frac{N}{2}-1$$

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

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

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

$$X\left(\frac{N}{2}+k\right) = X_1(k) + W_N^{\frac{N}{2}+k} X_2(k) = X_1(k) - W_N^k X_2(k) \qquad 0 \le k \le \frac{N}{2}-1$$

归纳起来有

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

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

$(N/2)^2$ $(N/2)$ $(N/2)^2$

**乘法计算量: $N^2/2+N/2$** **$N^2 \rightarrow N^2/2+N/2$**

可见,

$$N-DFT \rightarrow \begin{cases} \frac{N}{2}-DFT \\ \frac{N}{2}-DFT \end{cases} \rightarrow N-DFT$$

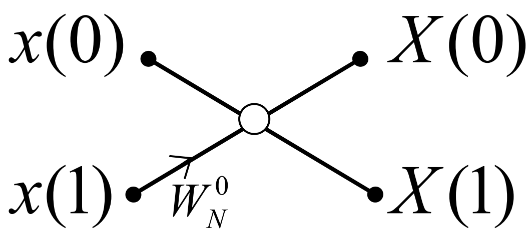
2-DFT：

$$X(k)=\sum_{n=0}^{1}x(n)W_2^{kn}$$

$$=x(0)W_2^0+x(1)W_2^k \qquad k=0,1$$

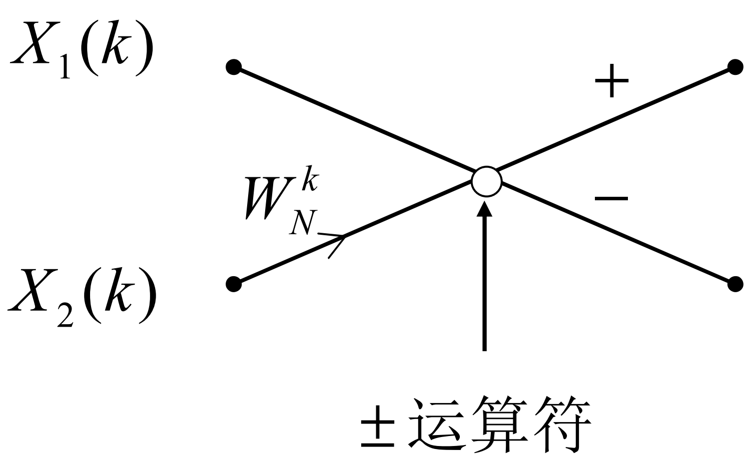
$$X(0)=x(0)+x(1)=x(0)+W_N^0x(1)$$

$$X(1)=x(0)-x(1)=x(0)-W_N^0x(1)$$




可见仅需计算“+/-”运算。

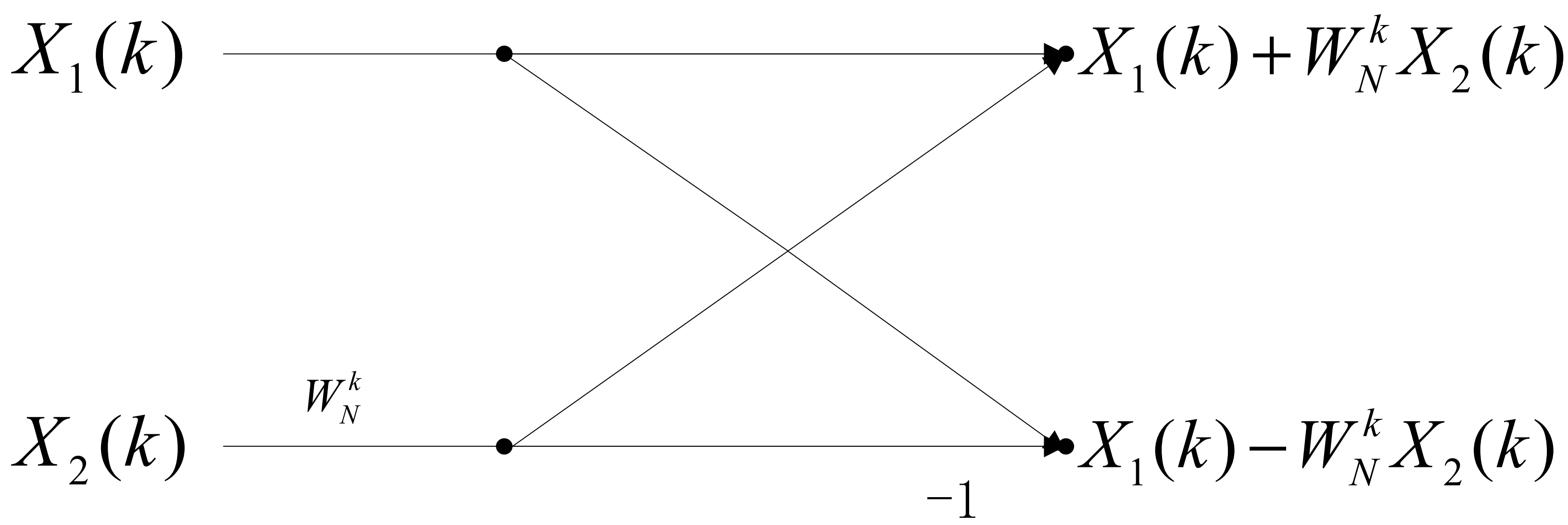
**上述运算可用下列蝶形信号流图表示:**

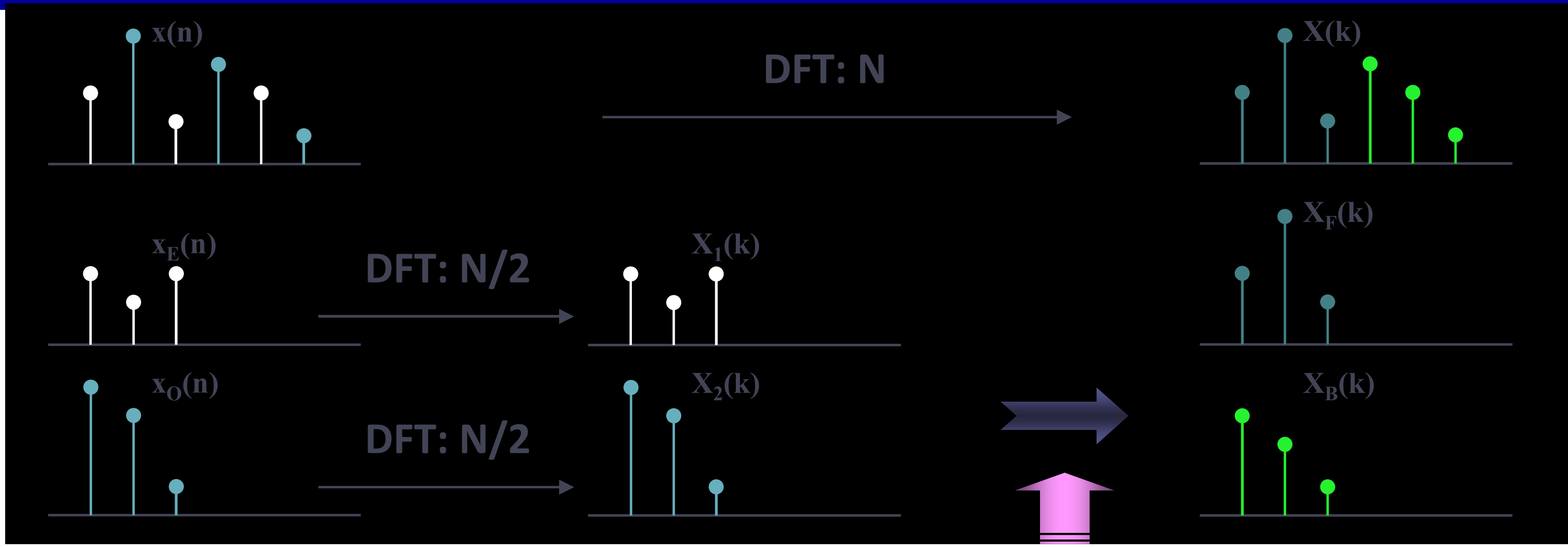


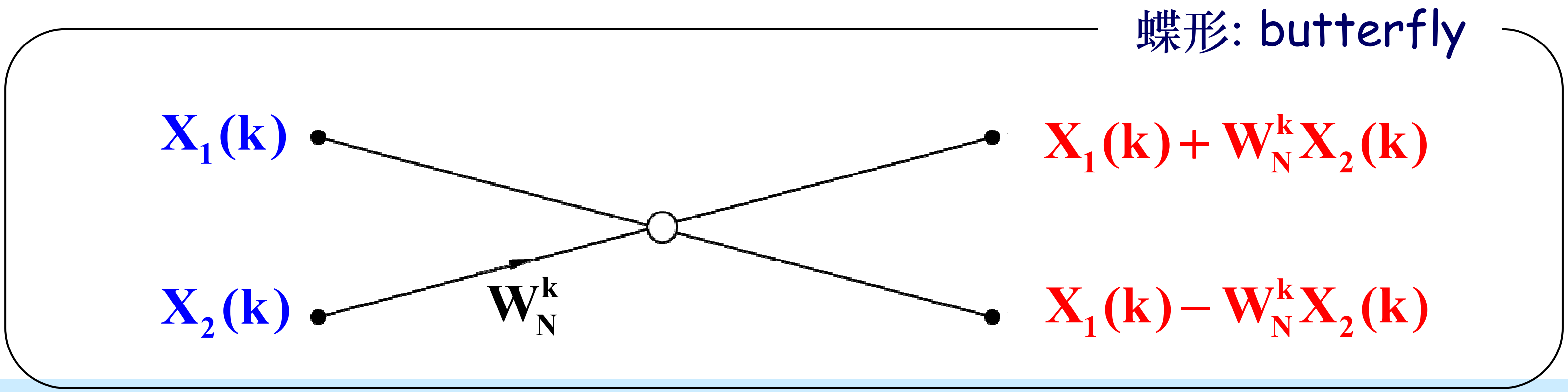

$$X(k)=X_1(k)+W_N^k X_2(k)$$

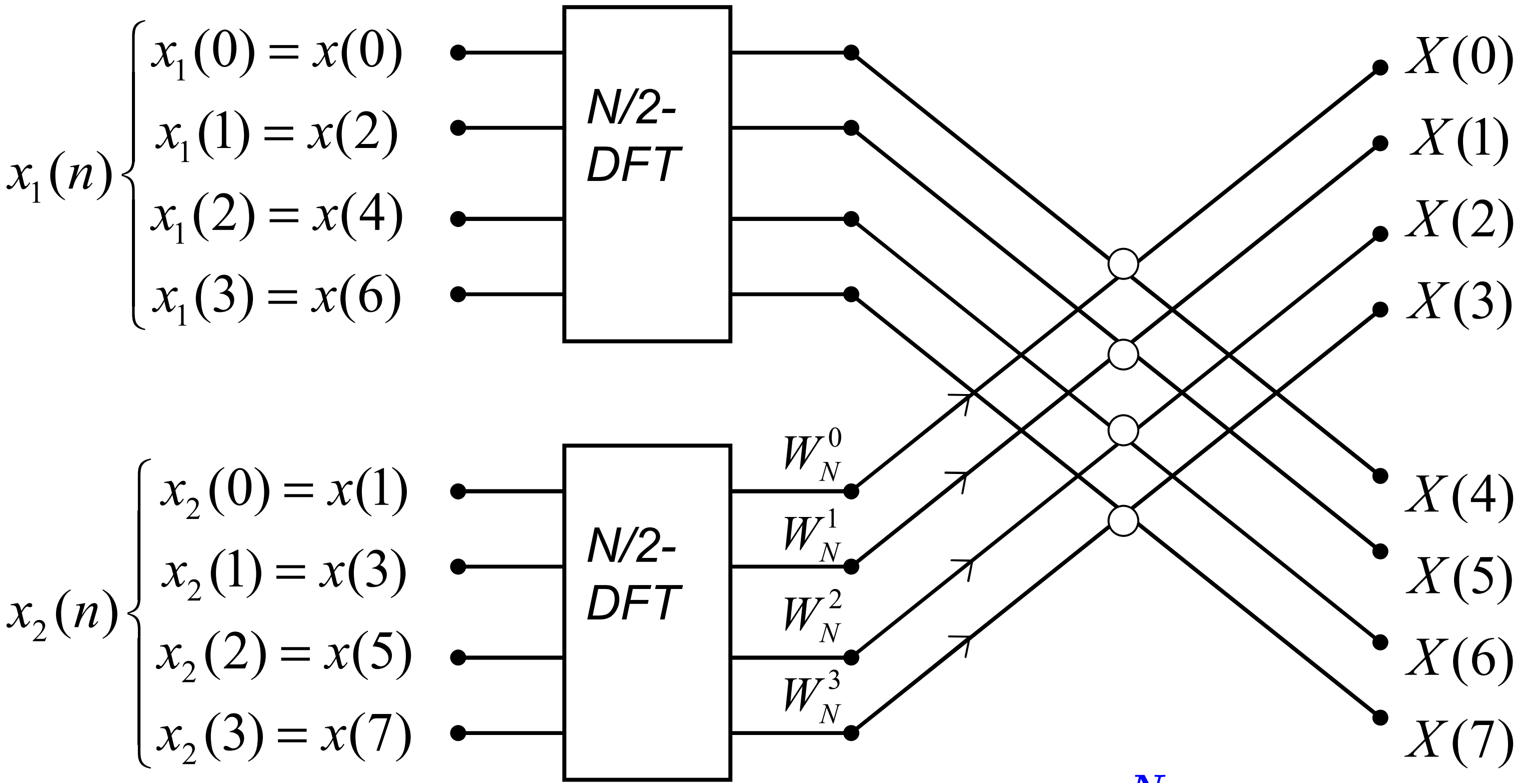
$$X\left(\frac{N}{2}+k\right)=X_1(k)-W_N^k X_2(k)$$

图 4-1 蝶形运算流图符号


$X_1(k)$
$X_1(k)+W_N^k X_2(k)$
$X_2(k)$
$W_N^k$
$X_1(k)-W_N^k X_2(k)$
$-1$


x(n)
DFT: N
X(k)
$x_E(n)$
DFT: N/2
$X_1(k)$
$X_F(k)$
$x_O(n)$
$X_2(k)$
DFT: N/2
$X_B(k)$


蝶形: butterfly
$X_1(k)$
$X_2(k)$
$W_N^k$
$X_1(k)+W_N^kX_2(k)$
$X_1(k)-W_N^kX_2(k)$




$$W_N^k, \quad k = 0, 1, \ldots, \frac{N}{2} - 1$$

图4.2 按时间抽取，将一个N点DFT分解为两个N/2点DFT

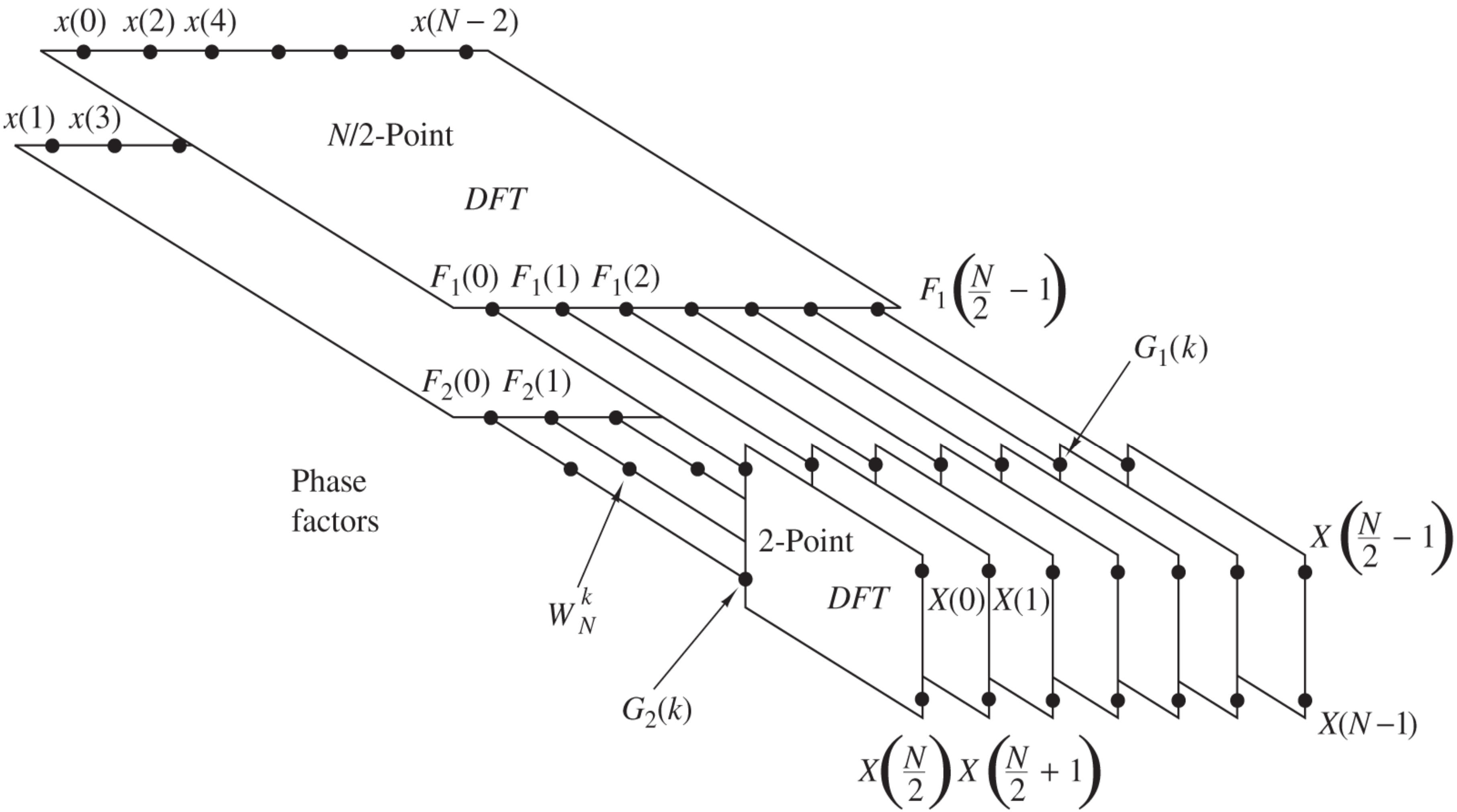
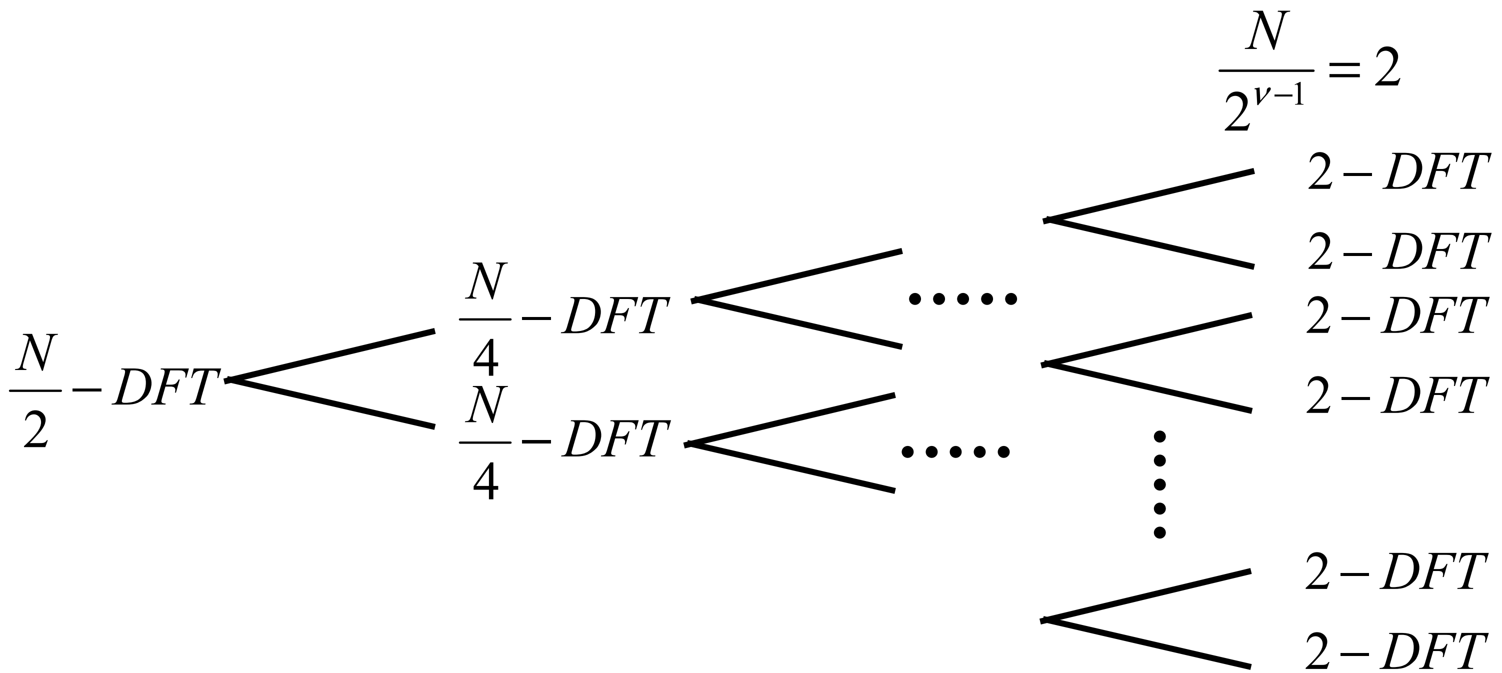



Figure 8.1.4 First step in the decimation-in-time algorithm.

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

$$\therefore \frac{N}{2} = 2^{\nu-1} = 2 \times 2^{\nu-2}$$



## 每个N/2 点子序列进一步分解为两个N/4点子序列

- 偶序列中的偶数序列
- 偶序列中的奇数序列
- 奇序列中的偶数序列
- 奇序列中的奇数序列

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

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

$$X_2(k) = X_5(k) + W_{N/2}^{k} X_6(k), \qquad k = 0,1,\ldots,\frac{N}{4}-1$$

$$X_2(k+\frac{N}{4}) = X_5(k) - W_{N/2}^{k} X_6(k), \qquad k = 0,1,\ldots,\frac{N}{4}-1$$

$$W_{N/2}^{k} = W_{N}^{2k}$$

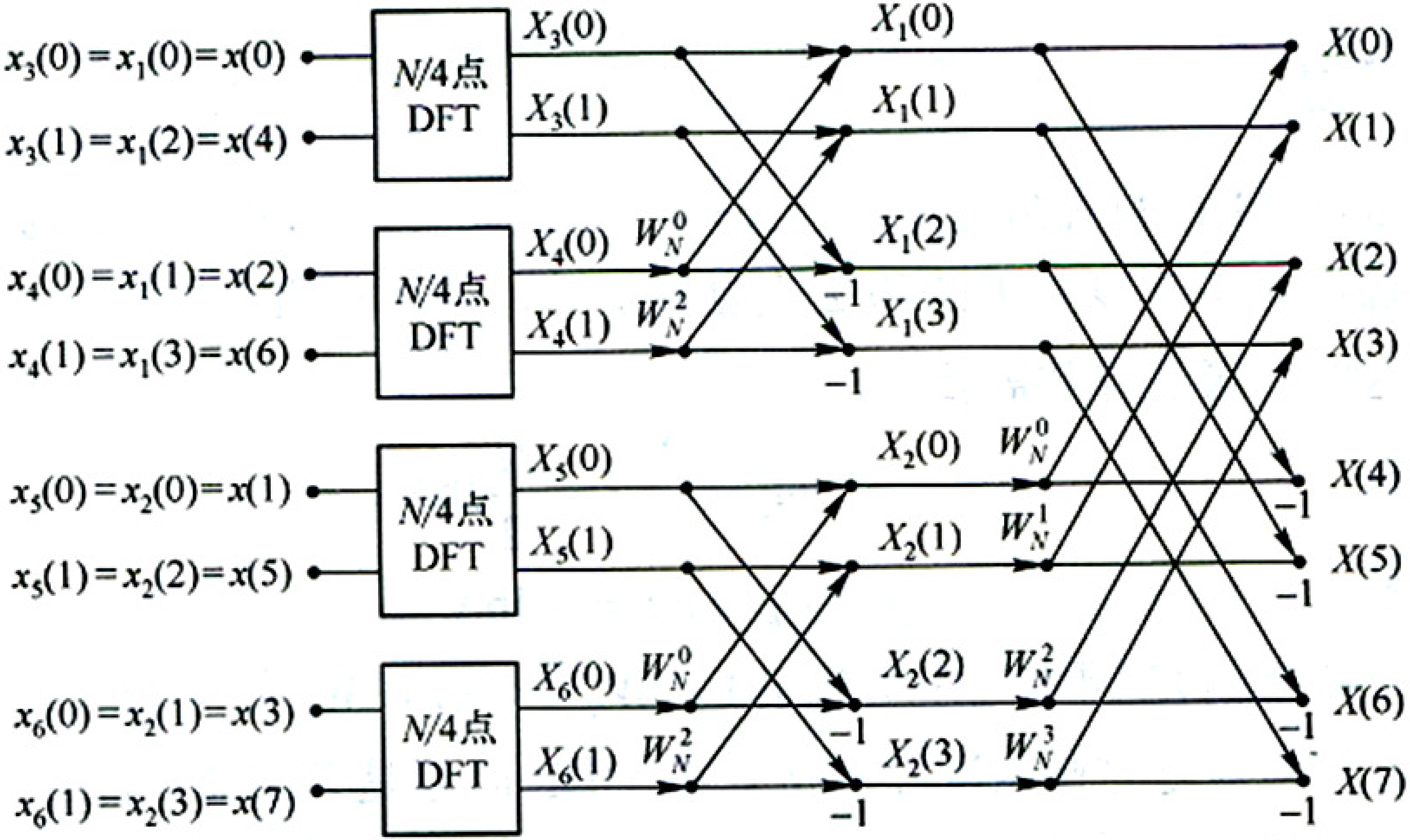
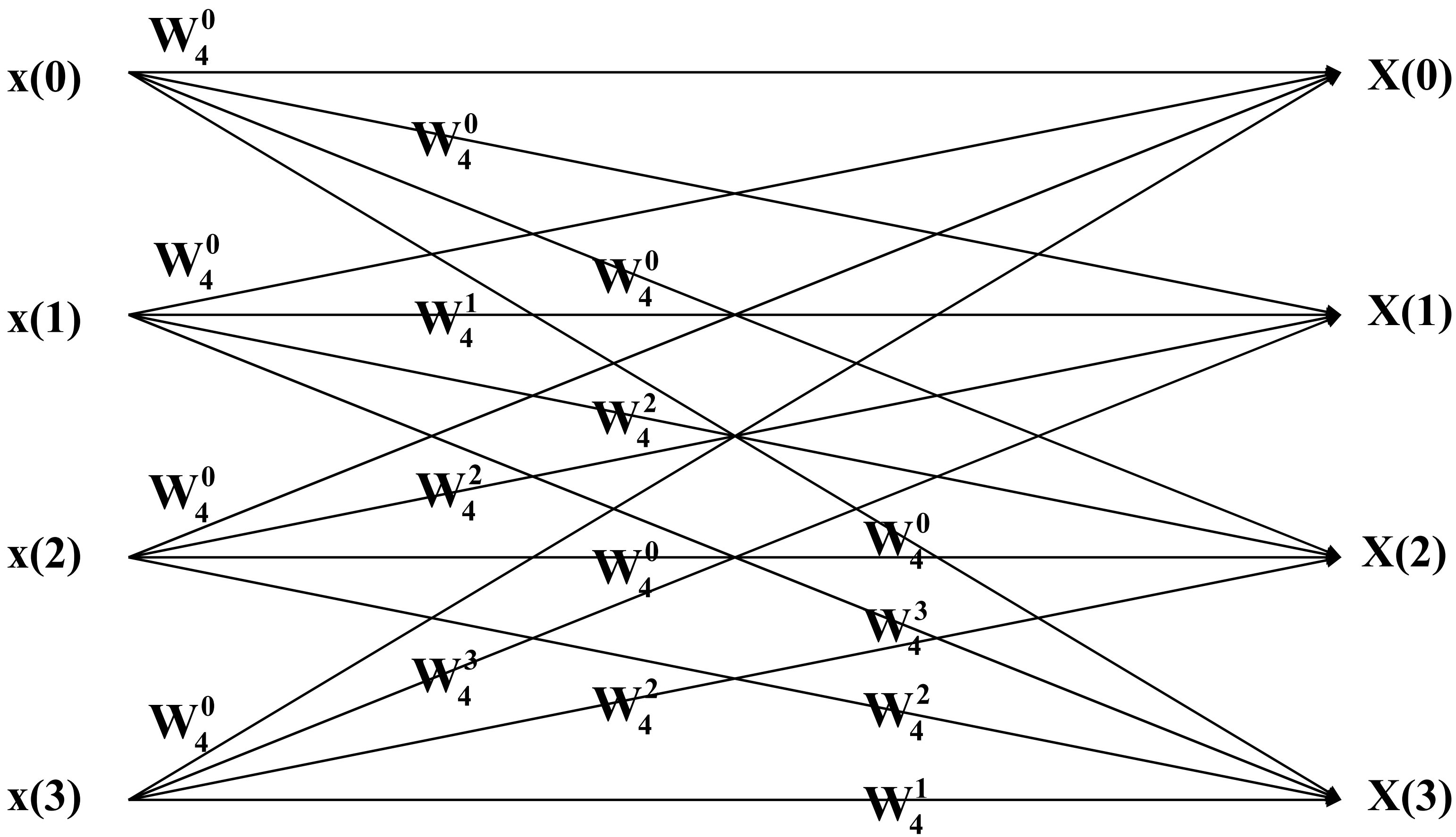
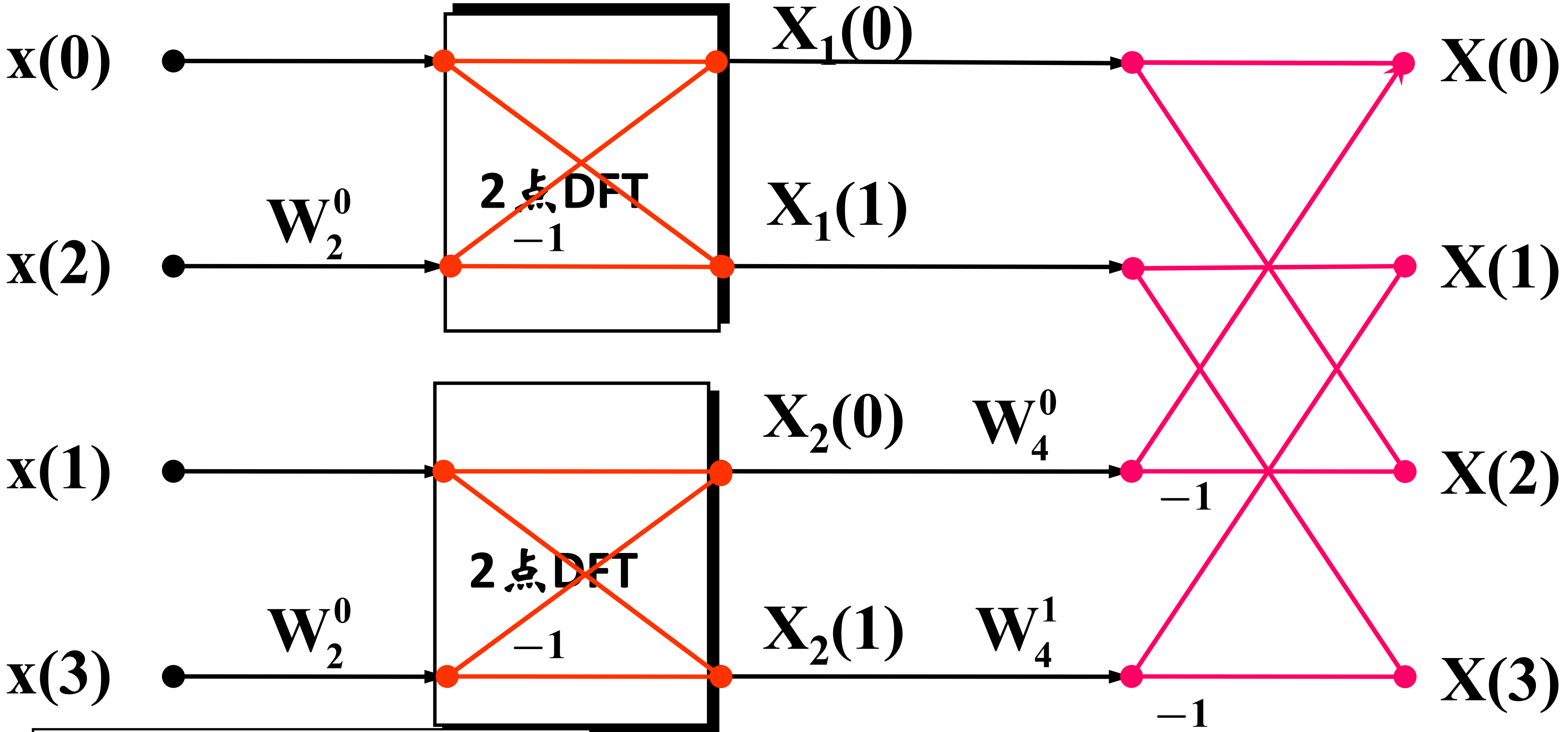



图 4.4 按时间抽选，将一个 $N$ 点 DFT 分解为四个 $N/4$ 点 DFT（$N=8$）

## See how the direct DFT is operated:




**4点DFT直接计算算法流图




观察：

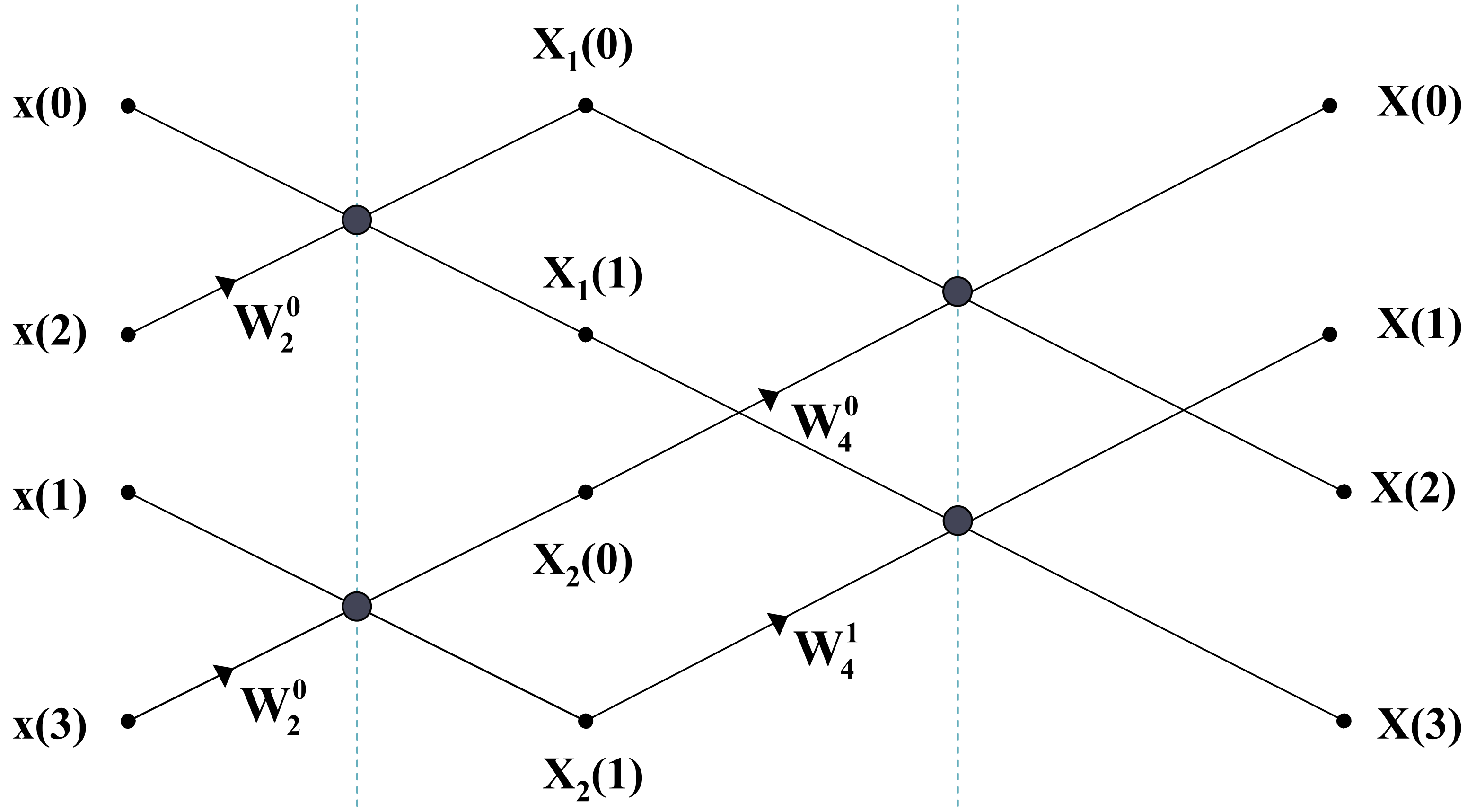
- 有几级？
- 每级有几个互异旋转因子？
- 每级有几个蝶形？

$$X(k) = X_1(k) + W_4^k X_2(k), \ k = 0,1$$

$$X(k+2) = X_1(k) - W_4^k X_2(k), \ k = 0,1$$

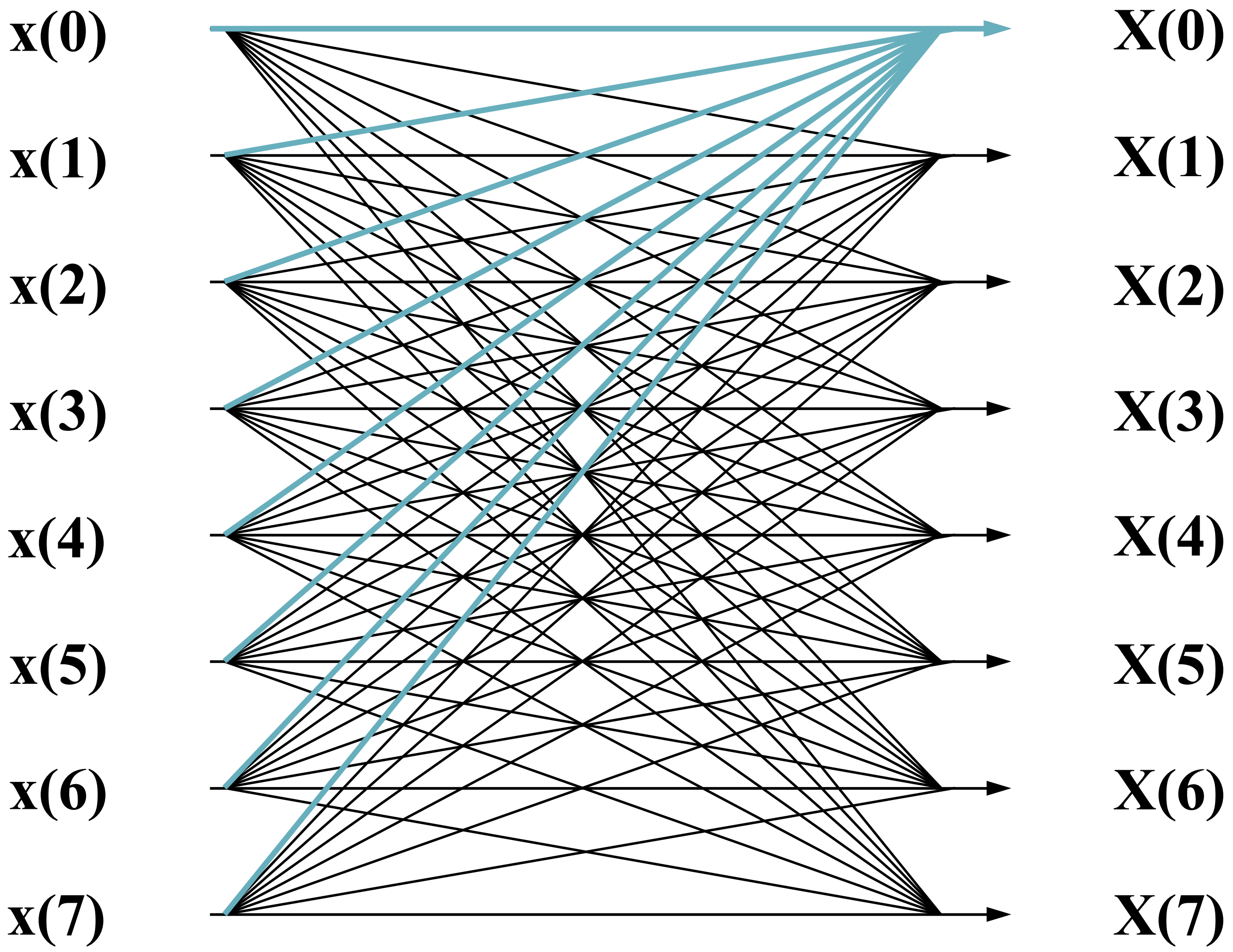
**4点基2时间抽取FFT算法流图

对于 DIT-FFT 而言，输入序列为非顺序，而输出序列为正常顺序，
各级具有相同数目的蝶形数，但参加蝶形运算的数据距离从左向右逐渐变远

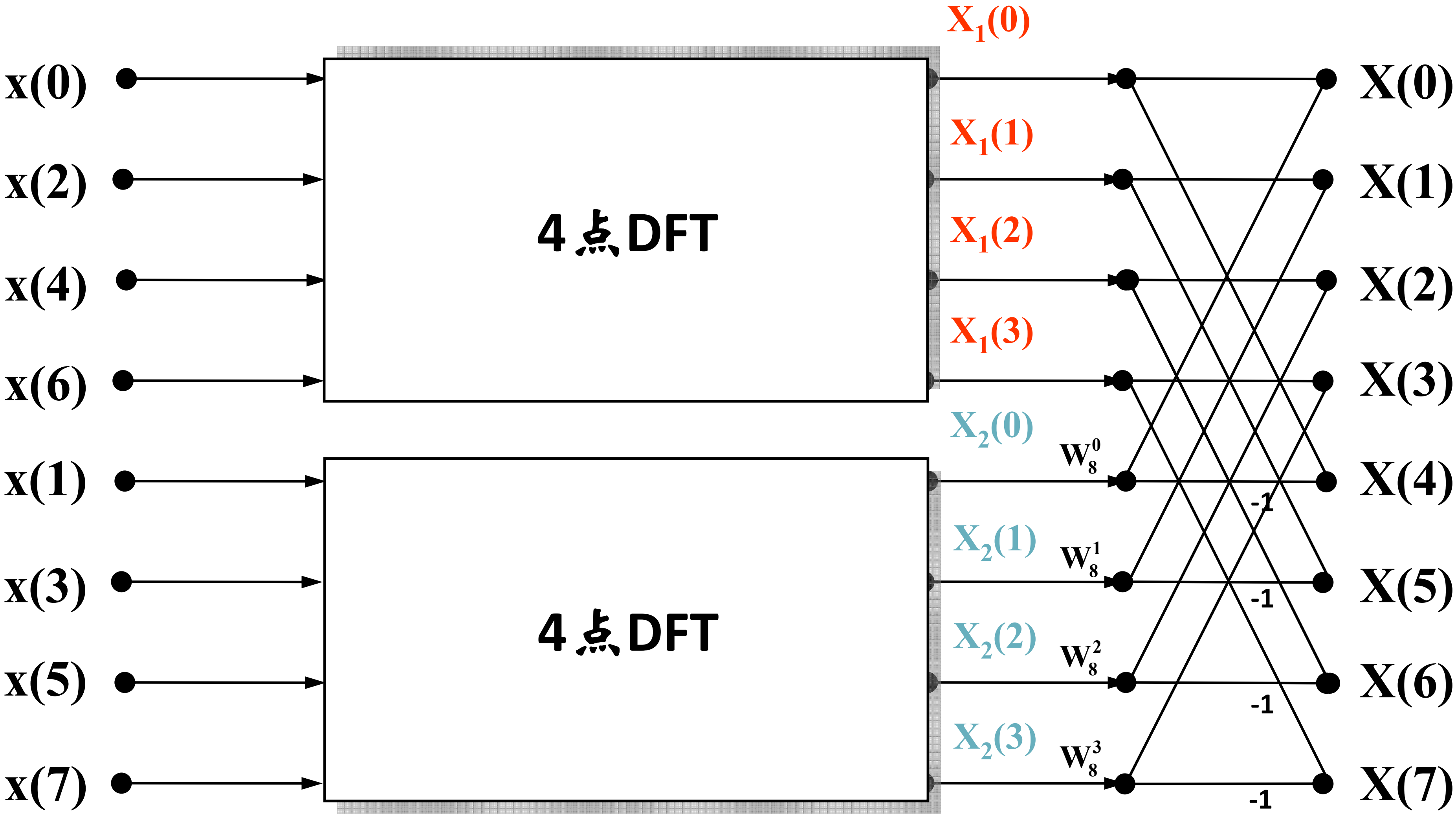





**4点基2时间抽取FFT算法流图



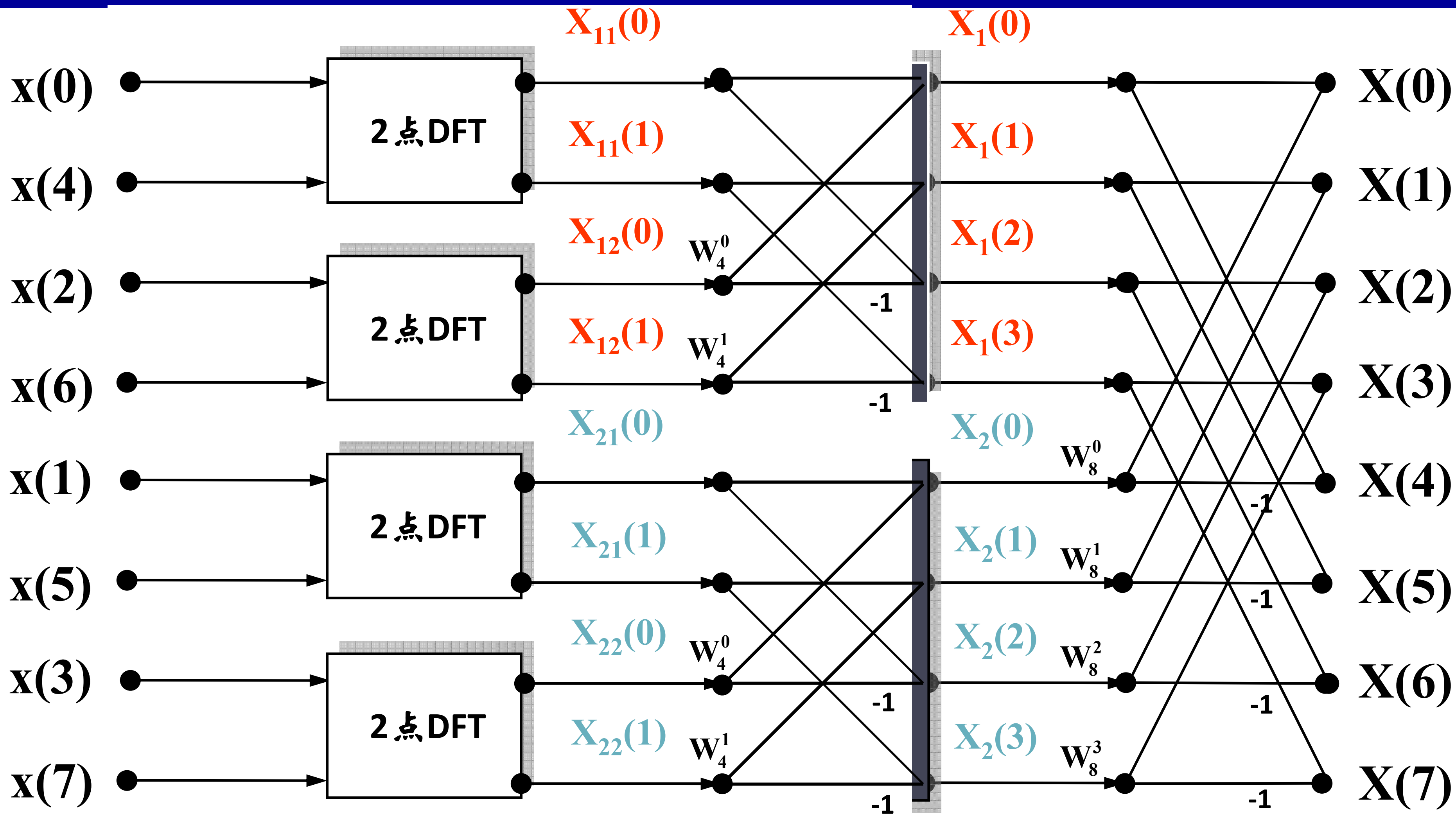

**8点DFT直接计算算法流图




$$X(k) = X_1(k) + W_8^k X_2(k),\ k = 0,1,2,3$$

$$X(k+4) = X_1(k) - W_8^k X_2(k),\ k = 0,1,2,3$$

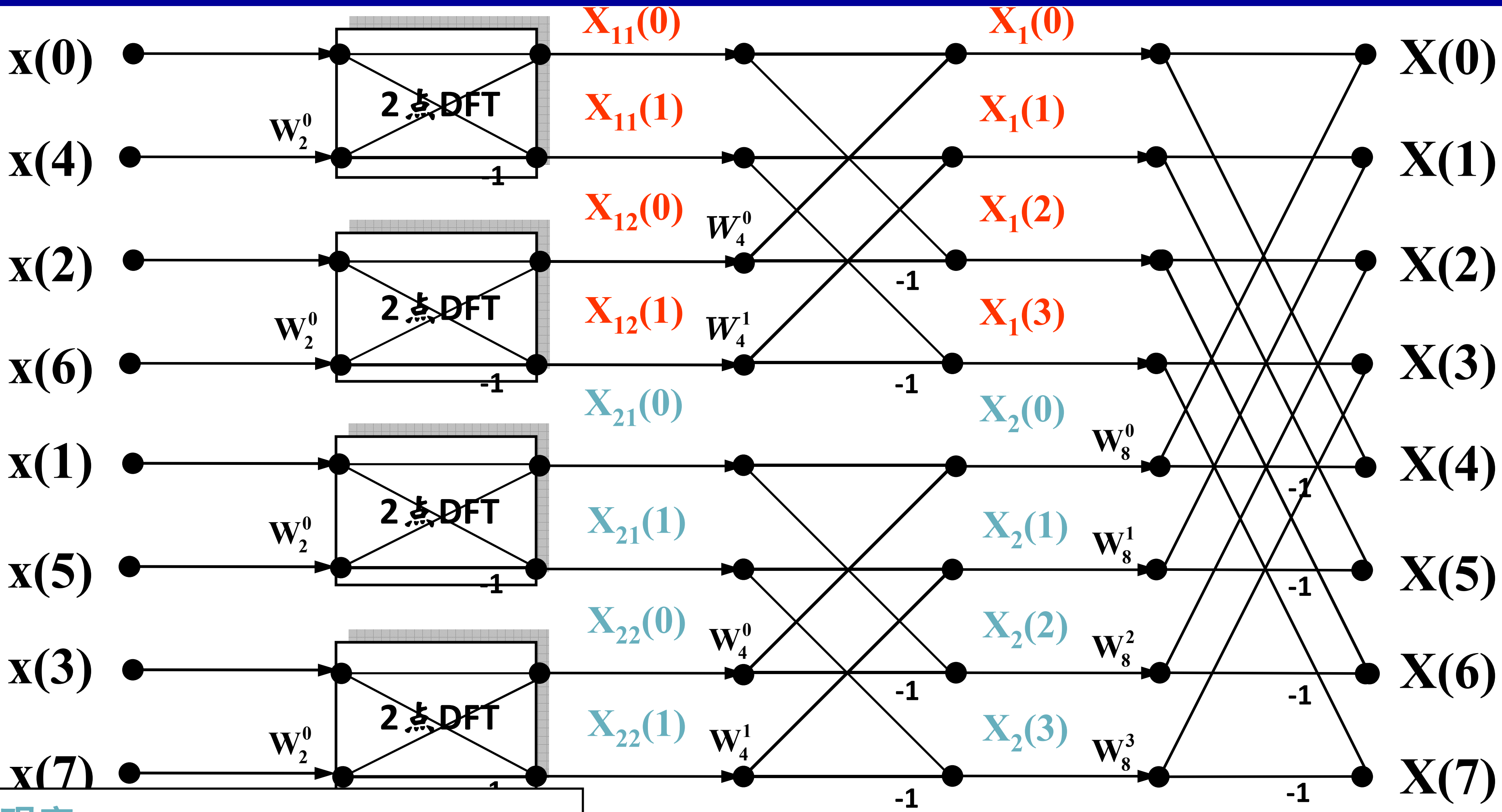
**8点基2时间抽取FFT算法流图




$$X(k) = X_1(k) + W_8^k X_2(k), \; k = 0,1,2,3$$

$$X(k+4) = X_1(k) - W_8^k X_2(k), \; k = 0,1,2,3$$

**8点基2时间抽取FFT算法流图

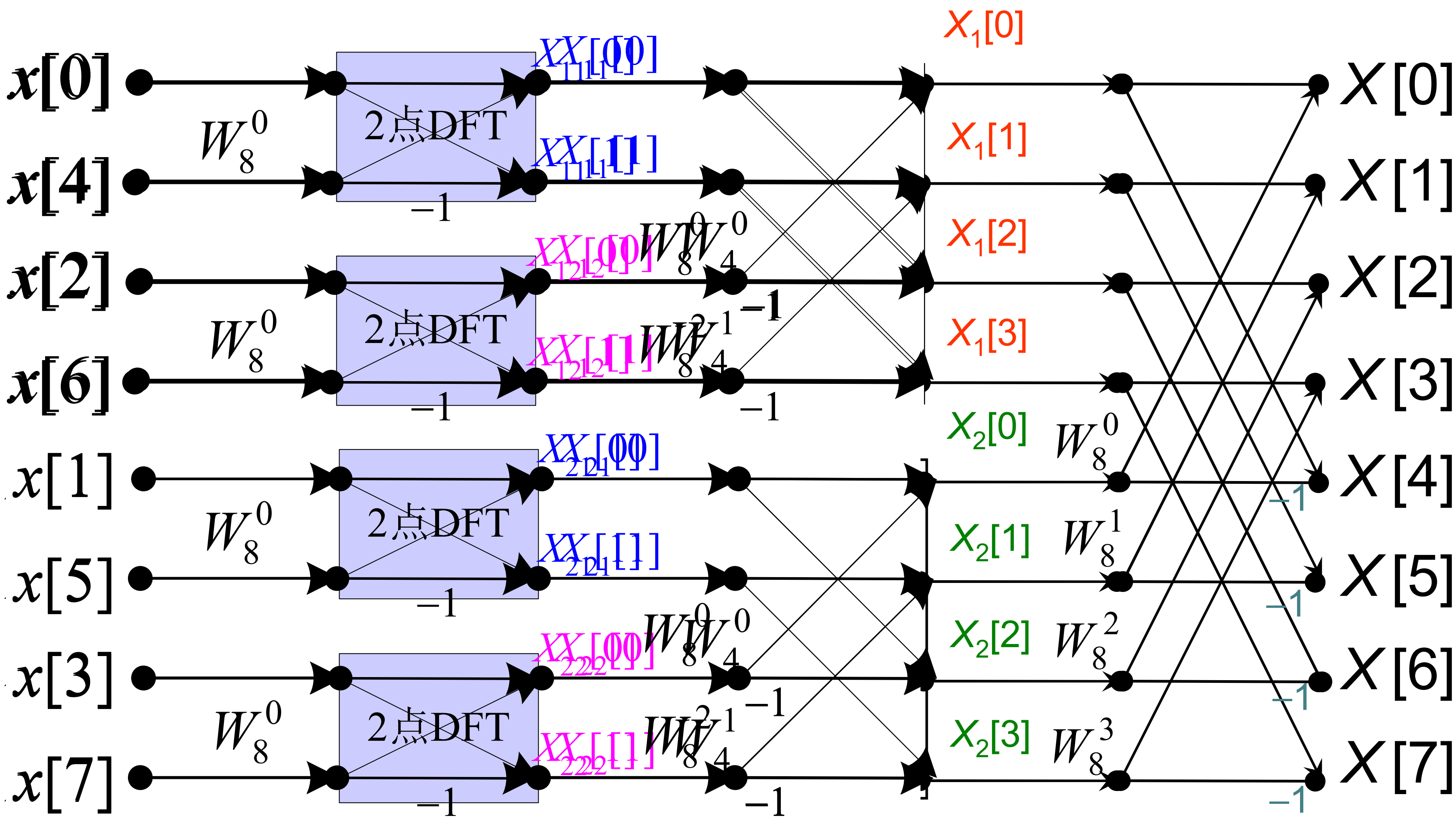



观察：

- 有几级？
- 每级有几个互异旋转因子？
- 每级有几个蝶形？

$$\overline{X}(k) = \sum_{n=0}^{1} \overline{x}(n)e^{-j\pi nk}, \quad k = 0, 1$$

**8点基2时间抽取FFT算法流图**




8点基2时间抽取FFT算法流图

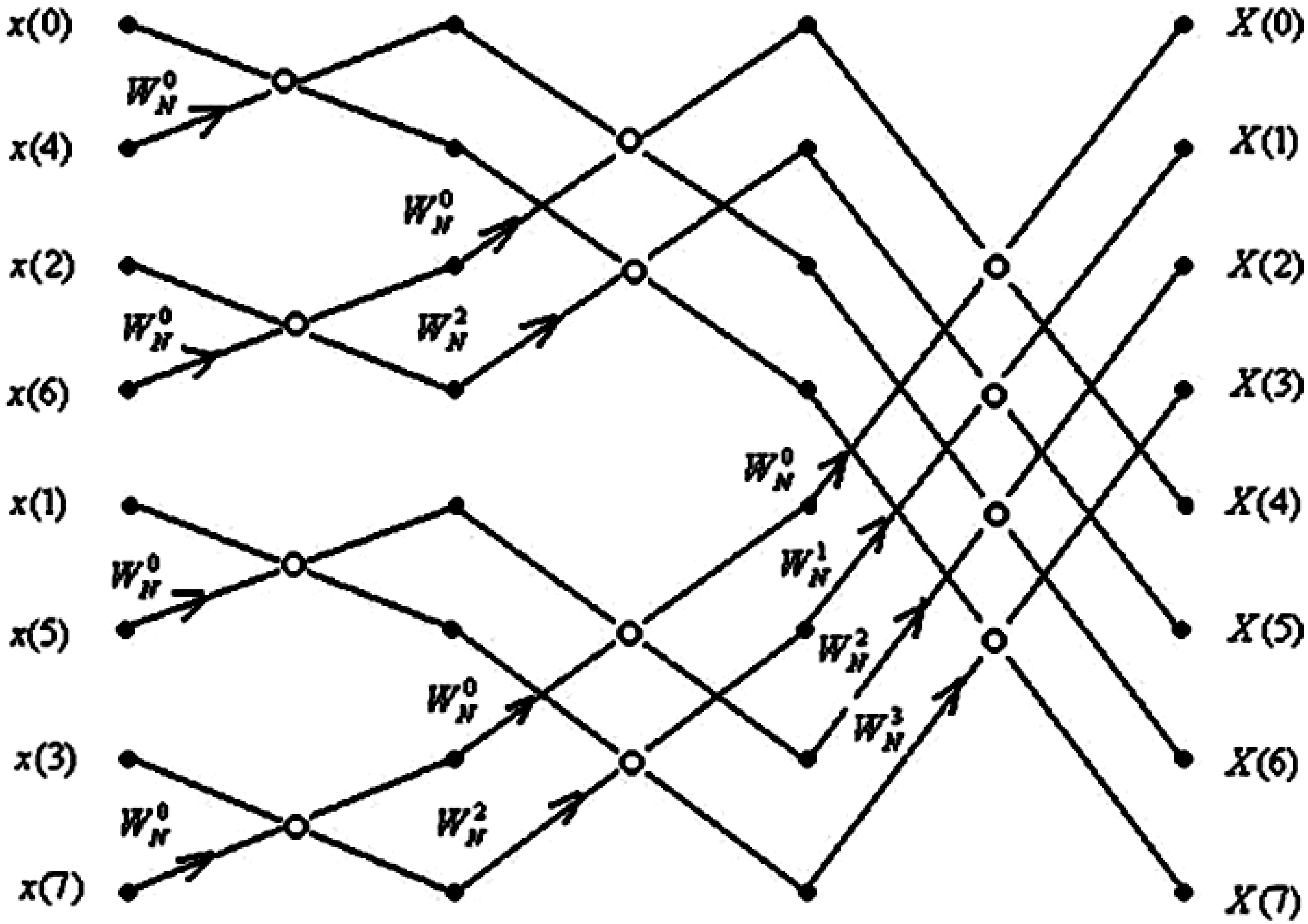



图4-5 N=8时的按时间抽取FFT运算流图

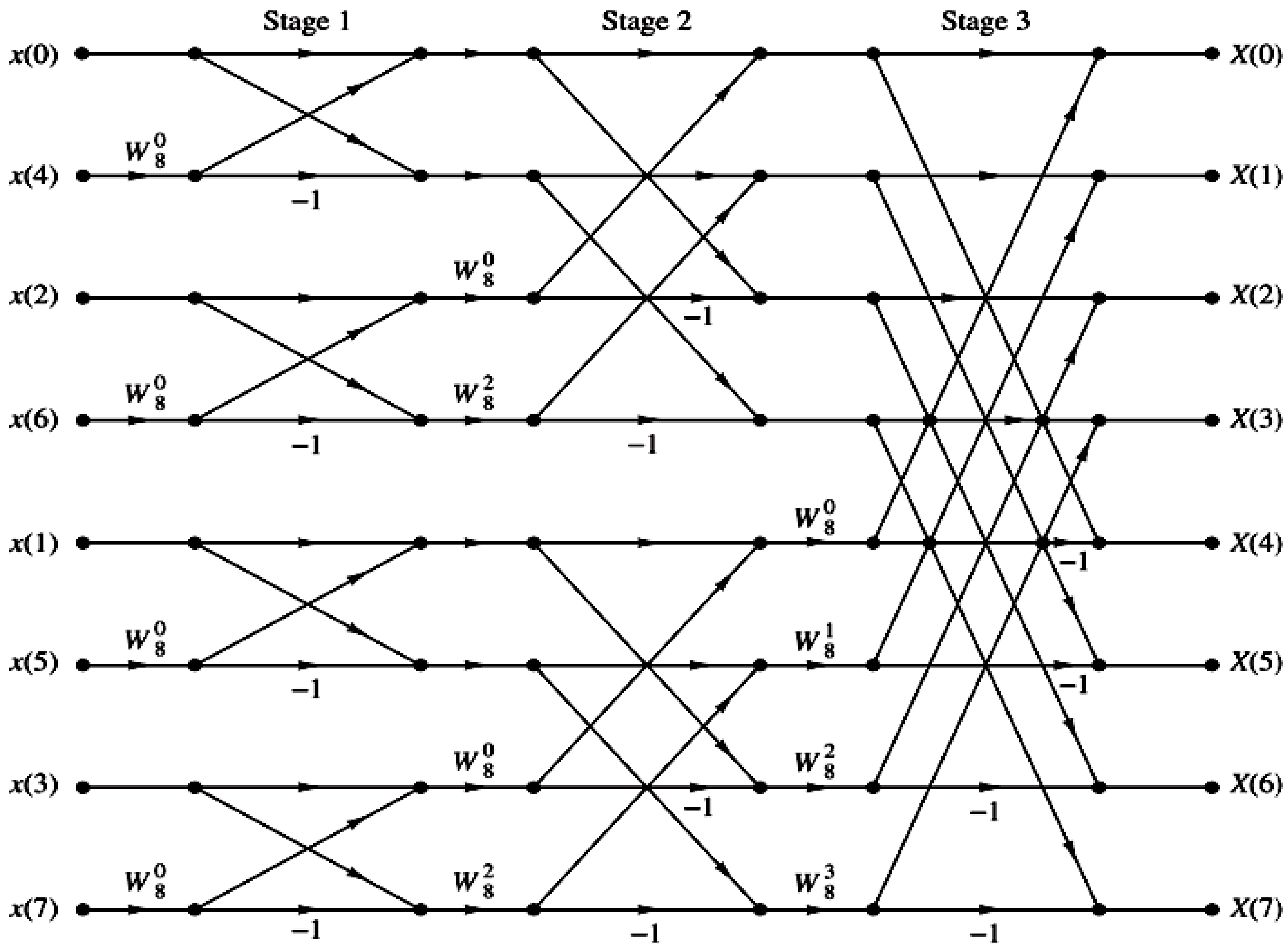



**Figure 8.1.6** Eight-point decimation-in-time FFT algorithm.

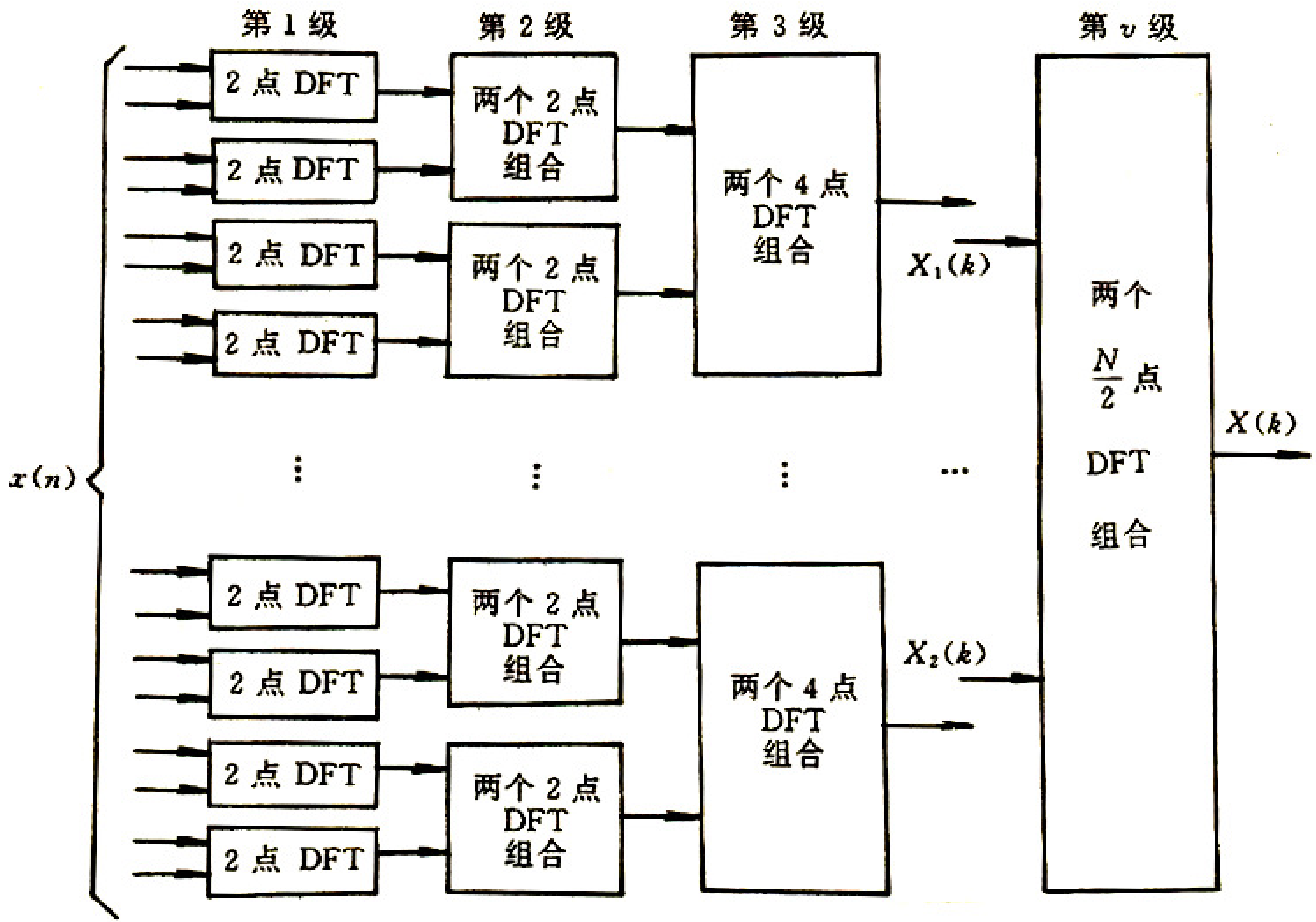



图4-6 N点基-2FFT的 v 级迭代过程

# 二、运算量比较

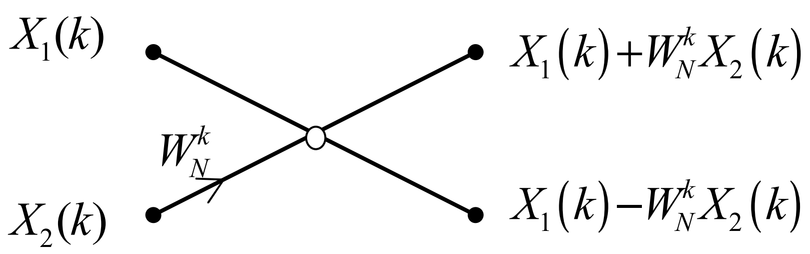
**1.DIT-FFT：N=2 ^v^**

$N-DFT \rightarrow v$ 级分解/蝶形运算　　$v=\log_2 N$

每一级：均有 $\dfrac{N}{2}$ 蝶形运算

$$* \quad : \quad \frac{N}{2}$$

$$+ \quad : \quad N$$




所有 $v$ 级：

$$\left.\begin{aligned} * \quad &: \quad \frac{N}{2}\times v=\frac{N}{2}\log_2^N \\ + \quad &: \quad N\times v=N\log_2^N \end{aligned}\right\}$$

## 2. DFT

$$\left.\begin{array}{l} * \quad : \quad N^2 \\ + \quad : N(N-1) \end{array}\right\} \sim N^2$$

## 3. DIT-FFT的运算效率

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

表 4.1　FFT 算法与直接 DFT 算法的比较

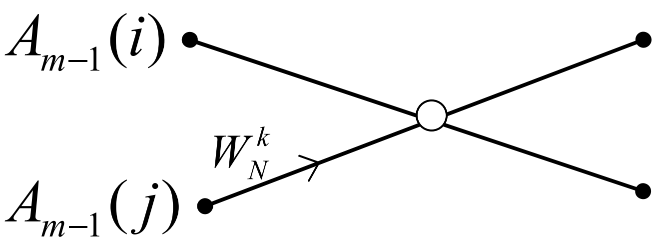
| $N$ | $N^2$ | $\frac{N}{2}\mathrm{lb}N$ | $N^2/\left(\frac{N}{2}\mathrm{lb}N\right)$ |
|---|---|---|---|
| 2 | 4 | 1 | 4.0 |
| 4 | 16 | 4 | 4.0 |
| 8 | 64 | 12 | 5.4 |
| 16 | 256 | 32 | 8.0 |
| 32 | 1 024 | 80 | 12.8 |
| 64 | 4 096 | 192 | 21.4 |
| 128 | 16 384 | 448 | 36.6 |
| 256 | 65 536 | 1 024 | 64.0 |
| 512 | 262 144 | 2 304 | 113.8 |
| 1024 | 1 048 576 | 5 120 | 204.8 |
| 2048 | 4 194 304 | 11 264 | 372.4 |

# 三、DIT-FFT算法的特点

规律分析

## 1.　原位运算(In-place)

**每列计算可在原位进行，共需N个复数存储器，节省存储单元。**



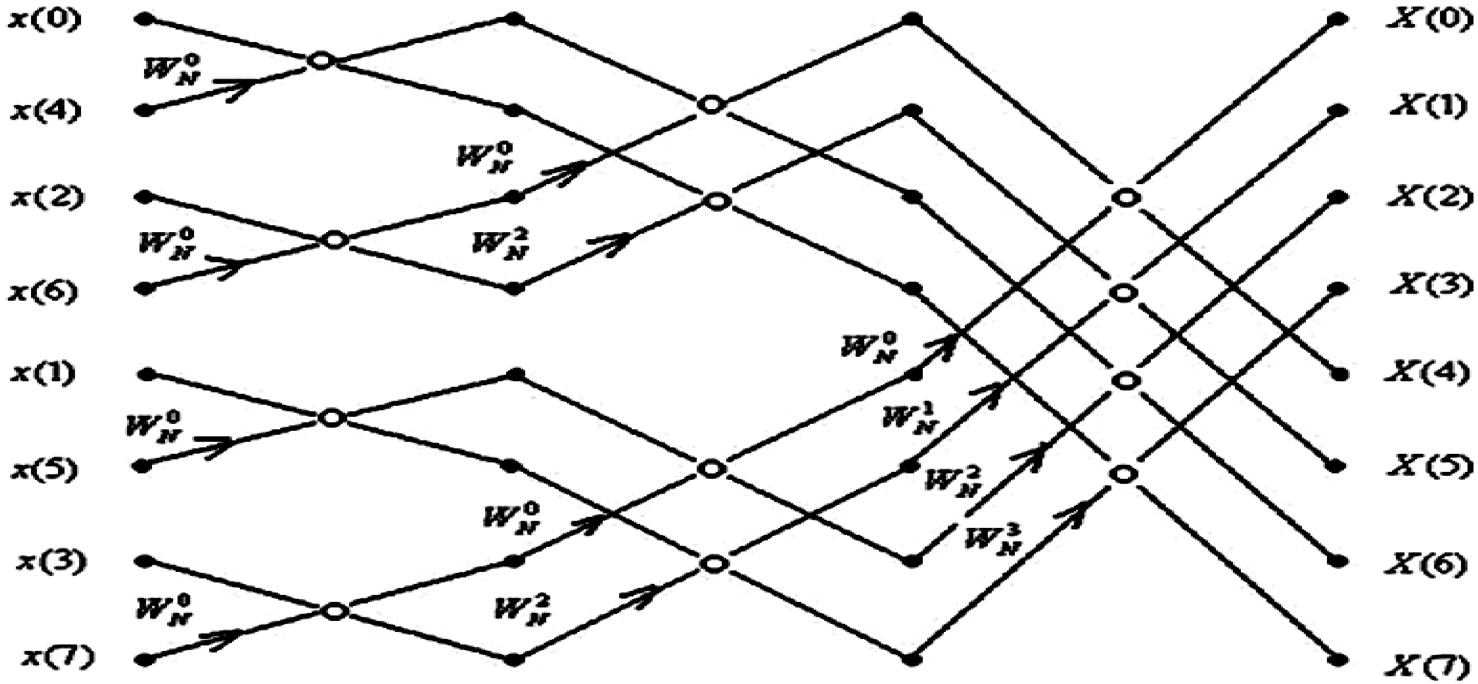

$$A_m(i) = A_{m-1}(i) + W_N^k A_{m-1}(j)$$

$$A_m(j) = A_{m-1}(i) - W_N^k A_{m-1}(j)$$

$m$ — 第$m$列迭代

$i, j$ — 数据所在的行数

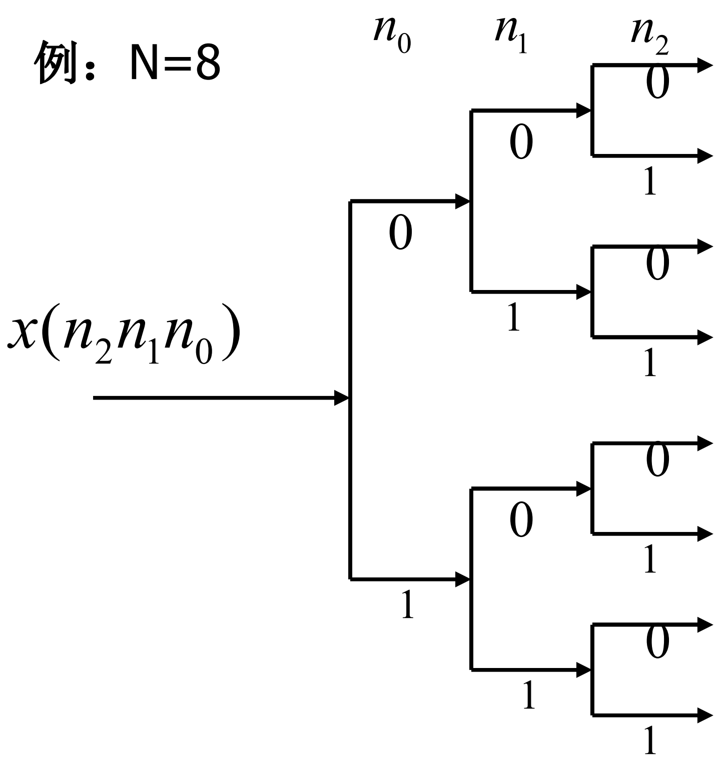
## 2. 输入序列的序号及整序规律




输入$x(n)$：乱序的
输出$X(k)$：顺序的

**(1) 乱序的原因：按时间抽取**

例：N=8




$x(000)=x(0)$

$x(100)=x(4)$

$x(010)=x(2)$

$x(110)=x(6)$

$x(001)=x(1)$

$x(101)=x(5)$

$x(011)=x(3)$

$x(111)=x(7)$

正好为DIT-FFT的输入

| $x(n)$ |
|---|
| $x(000)$ |
| $x(001)$ |
| $x(010)$ |
| $x(011)$ |
| $x(100)$ |
| $x(101)$ |
| $x(110)$ |
| $x(111)$ |

$\downarrow$ 比较

$$\underbrace{x(n_2n_1n_0)}_{\text{正序}} \rightarrow \underbrace{x(n_0n_1n_2)}_{\text{反序}}$$

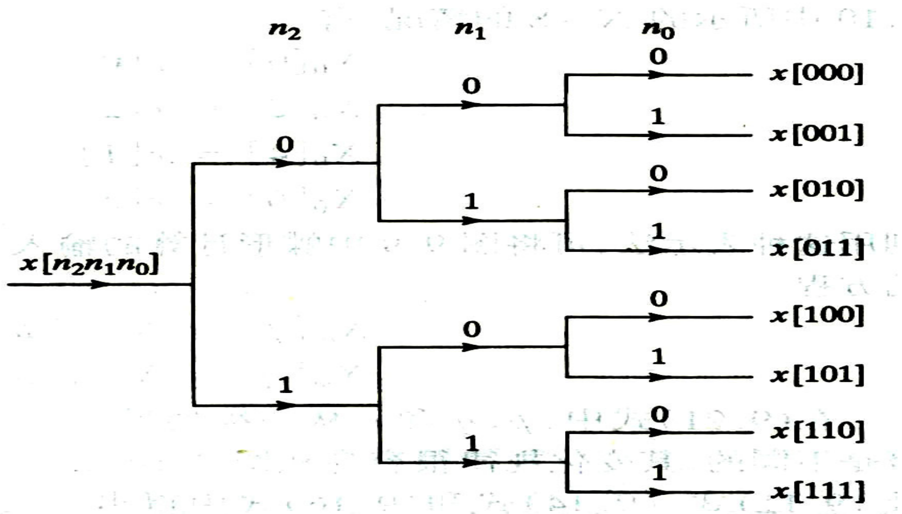
(2) 整序规律：倒序 (bit reverse order)

| 顺序 | | 倒序 | |
|---|---|---|---|
| 十进制数 $I$ | 二进制数 | 二进制数 | 十进制数 $J$ |
| 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 |

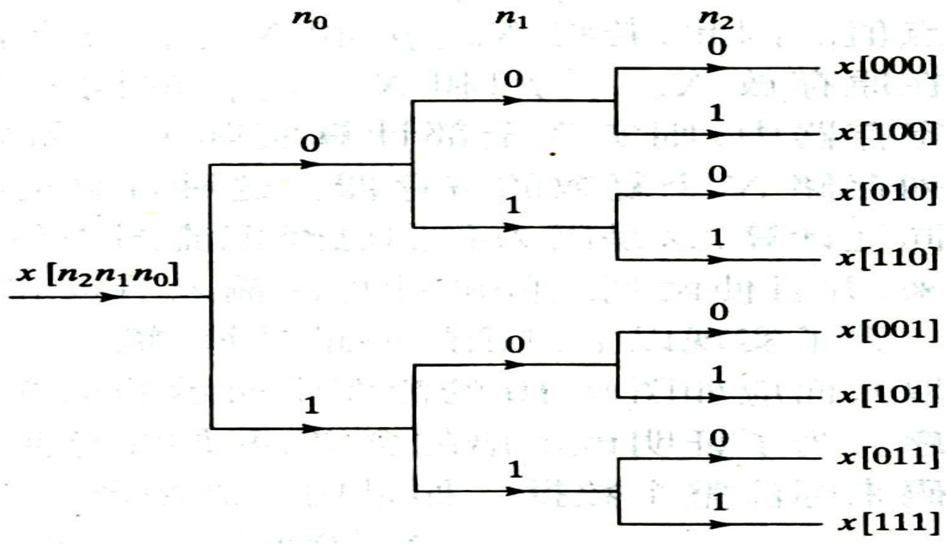
上一半0
下一半1

DIT-FFT 运算规律

正序


$n_2$
$n_1$
$n_0$
$x[n_2n_1n_0]$
$x[000]$
$x[001]$
$x[010]$
$x[011]$
$x[100]$
$x[101]$
$x[110]$
$x[111]$

倒序


$n_0$
$n_1$
$n_2$
$x[n_2n_1n_0]$
$x[000]$
$x[100]$
$x[010]$
$x[110]$
$x[001]$
$x[101]$
$x[011]$
$x[111]$

## 3. 蝶形运算两点间的距离和$W_N^k$的变化规律

距离

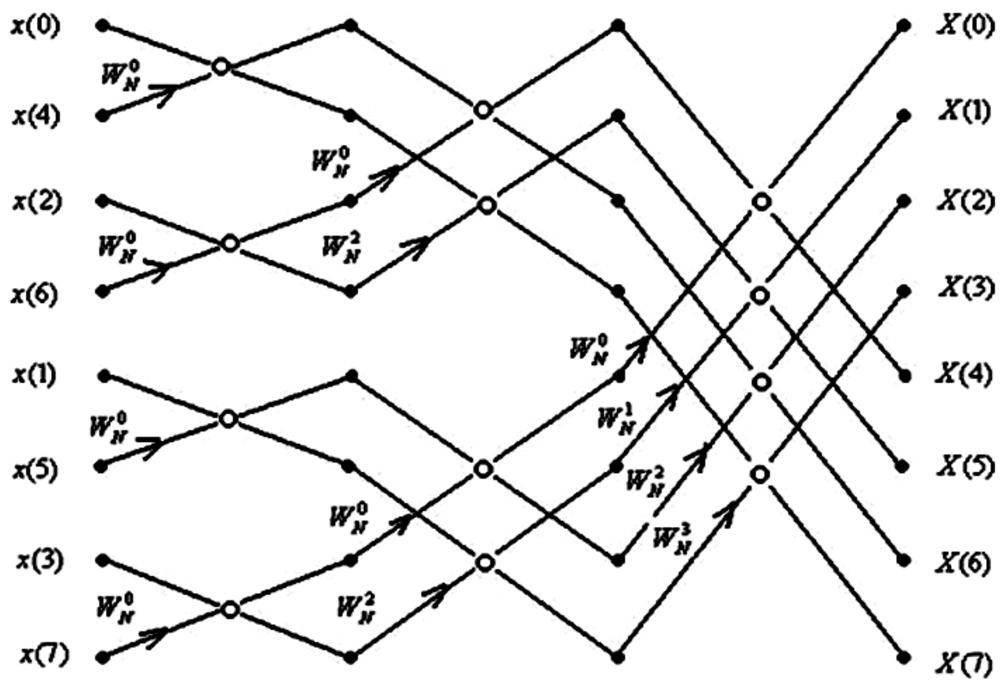
列 1: 1

列 2: 2

列 3: 4

列 m: $2^{m-1}$

列 v: N/2




$$W_N^r$$

列v: $W_N^0, \quad W_N^1, \quad \ldots W_N^{\frac{N}{2}-1}$

列v-1: $W_N^0, \quad W_N^2, \quad W_N^4 \ldots$

列1: $W_N^0$

# 四、DIT-FFT算法的若干变体

对于任何流图，只要保持各节点所连的支路及其传输系数不变，则不论节点位置怎么排列，所得流图总是等效的，最后所得DFT结果也是正确的，只是数据的提取和存放次序不同而已.

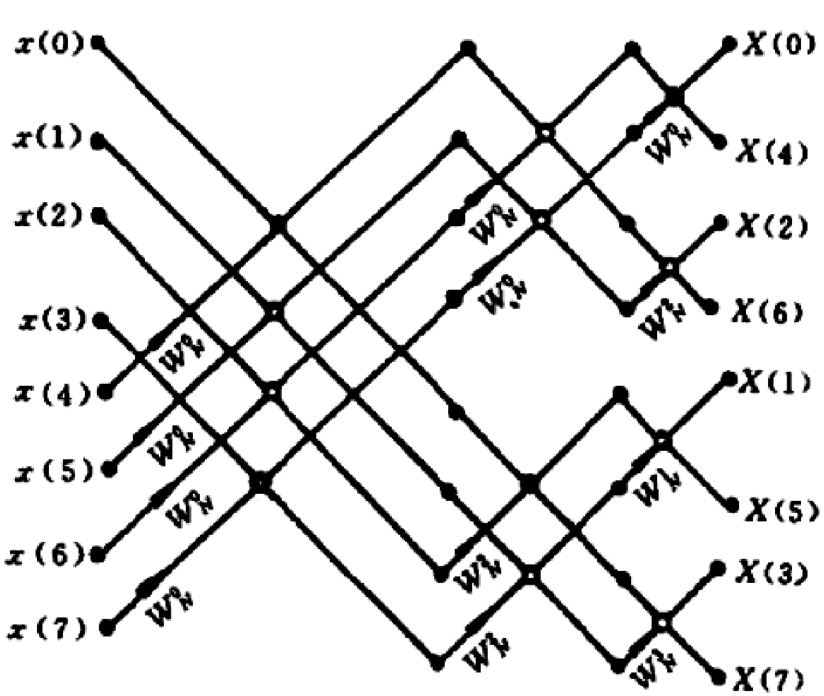


图 4—11 输入是自然顺序而输出是反序的流图

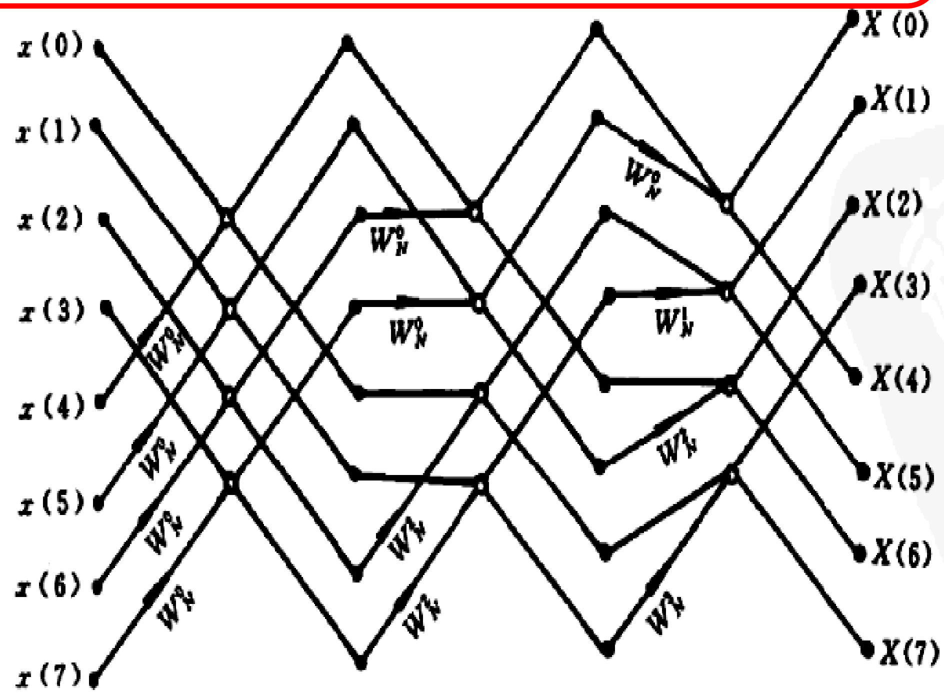


图 4—12 输入和输出都是自然顺序的流图

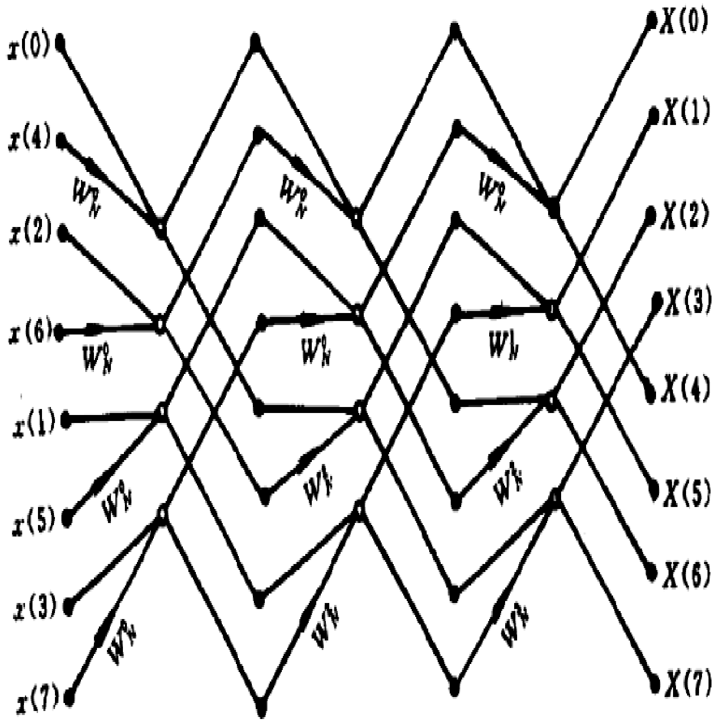



图 4-13 输入为反序,输出为自然顺序的恒定几何结构流图

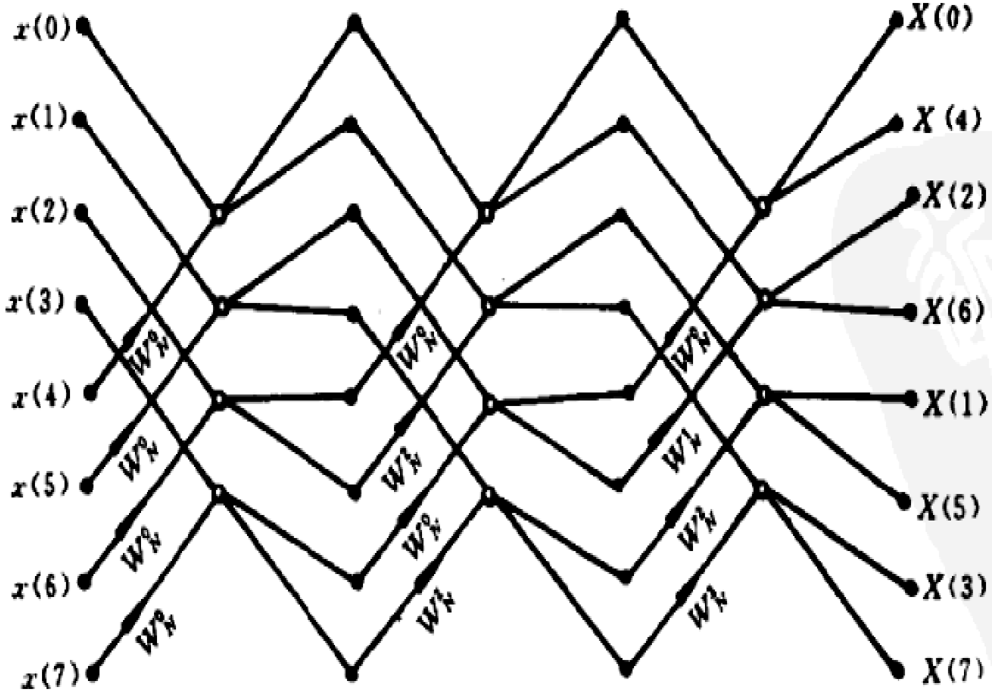



图 4-14 输入为自然顺序,输出为反序的恒定几何结构流图

# §4-4 按频率抽取(DIF)的FFT算法

## 一、算法原理 $N=2^{\nu}$

将$x(n)$，$0 \leqslant n \leqslant N-1$ 按顺序分为前后两半

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

$k=0,1,\ldots,N-1$

注意：$W_N=e^{-j\frac{2\pi}{N}} \neq W_{N/2}$

∴两个Σ和式并不是 N/2-DFT

由于 $\quad W_N^{k\frac{N}{2}} = e^{-j\frac{2\pi}{N}k\frac{N}{2}} = e^{-j\pi k} = (-1)^k$

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

$$= \sum_{n=0}^{\frac{N}{2}-1} [x(n) + (-1)^k x(n+\frac{N}{2})] W_N^{kn}, \qquad k = 0,1,\ldots,N-1$$

$$\because (-1)^k \begin{cases} 1 & k\text{为偶数} \\ -1 & k\text{为奇数} \end{cases}$$

$\therefore$ 可按$k$的奇偶取值将$X(k)$分为两部分：

## 将X(k)分成奇偶两部分

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

**N/2点DFT**

$$= \sum_{n=0}^{\frac{N}{2}-1} [x(n) + x(\frac{N}{2}+n)] W_{N/2}^{rn} \qquad r = 0,1,\ldots,\frac{N}{2}-1$$

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

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

**N/2点DFT**

$$= \sum_{n=0}^{\frac{N}{2}-1} [x(n) - x(\frac{N}{2}+n)] W_N^{n} \cdot W_{N/2}^{rn} \qquad r = 0,1,\ldots,\frac{N}{2}-1$$

显然，若令

$$x_1(n) \overset{\Delta}{=} [x(n)+x(n+\frac{N}{2})], \qquad n=0,1,\dots,\frac{N}{2}-1$$

$$X_1(k) \overset{\Delta}{=} X(2k), \qquad k=0,1,\dots,\frac{N}{2}-1$$

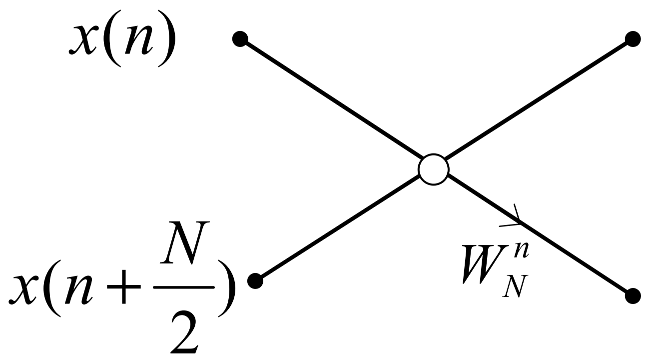
$$x_2(n) \overset{\Delta}{=} [x(n)-x(n+\frac{N}{2})]W_N^n, \qquad n=0,1,\dots,\frac{N}{2}-1$$

$$X_2(k) \overset{\Delta}{=} X(2k+1), \qquad k=0,1,\dots,\frac{N}{2}-1$$

$$X_1(k)=X(2k)=DFT[x_1(n)]=\sum_{n=0}^{\frac{N}{2}-1} x_1(n)W_{N/2}^{kn}, \quad k=0,1,\dots,\frac{N}{2}-1$$

$$X_2(k)=X(2k+1)=DFT[x_2(n)]=\sum_{n=0}^{\frac{N}{2}-1} x_2(n)W_{N/2}^{kn}, \quad k=0,1,\dots,\frac{N}{2}-1$$

$$N-DFT \xrightarrow[\text{按频率抽取}]{\text{按}k\text{为奇偶分解}} \begin{cases} \frac{N}{2}-DFT \\ \frac{N}{2}-DFT \end{cases}$$

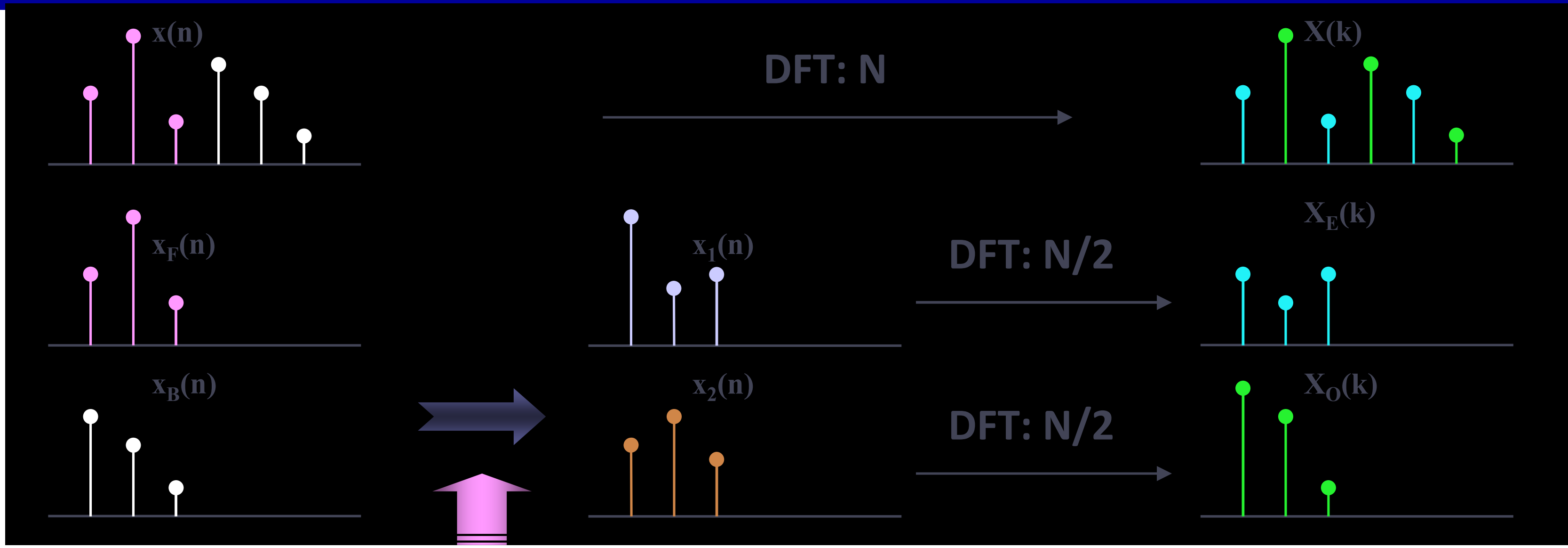


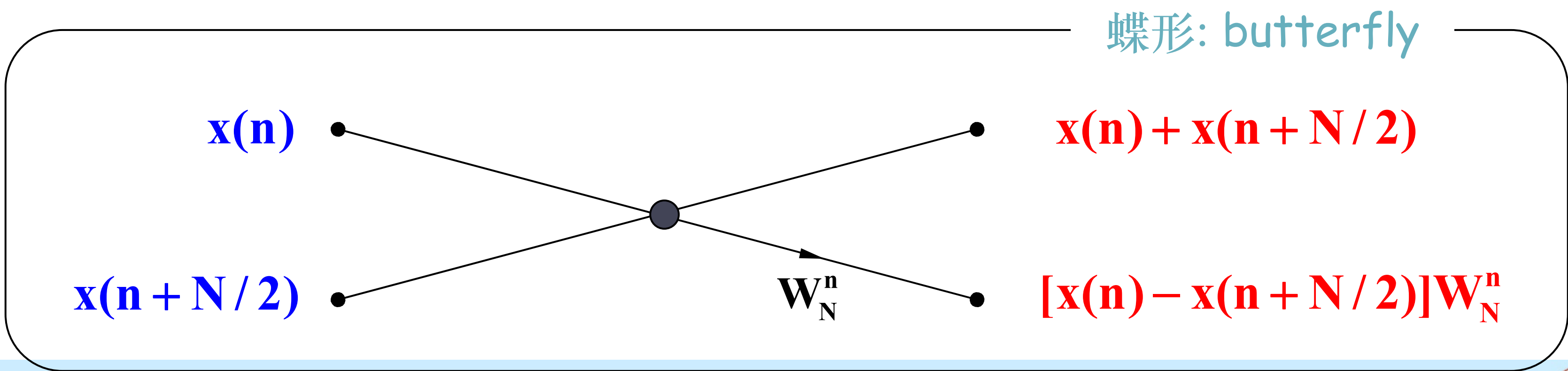

$$x_1(n)=x(n)+x(n+\frac{N}{2})$$

$$x_2(n)=[x(n)-x(n+\frac{N}{2})]W_N^n$$

$$n=0,1,\ldots,\frac{N}{2}-1$$

按频率抽取蝶形运算


x(n)
DFT: N
X(k)
$x_F(n)$
$x_1(n)$
DFT: N/2
$X_E(k)$
$x_B(n)$
$x_2(n)$
DFT: N/2
$X_O(k)$


蝶形: butterfly
$x(n)$
$x(n+N/2)$
$W_N^n$
$x(n)+x(n+N/2)$
$[x(n)-x(n+N/2)]W_N^n$

## 例：N=8，DIF的分解过程

$$\because N = 2^{\nu},\ \frac{N}{2} = 2^{\nu-1}\text{仍为偶数}(\nu \geq 3)$$

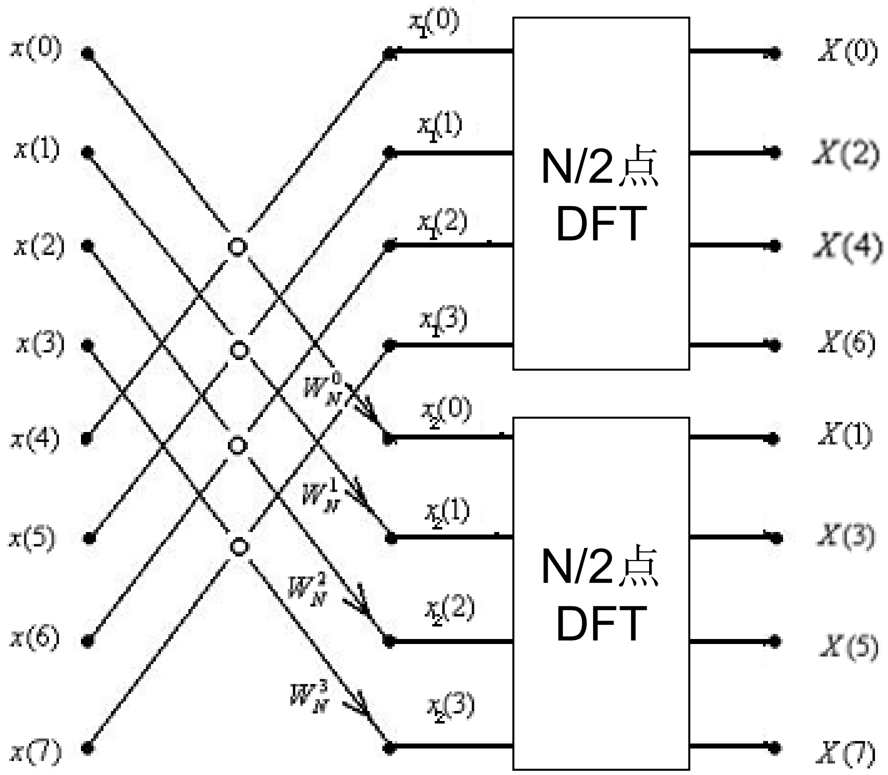
∴上述分解过程可继续下去，

直至分解 v 次/步后变成求

N/2 个 2-DFT 为止。

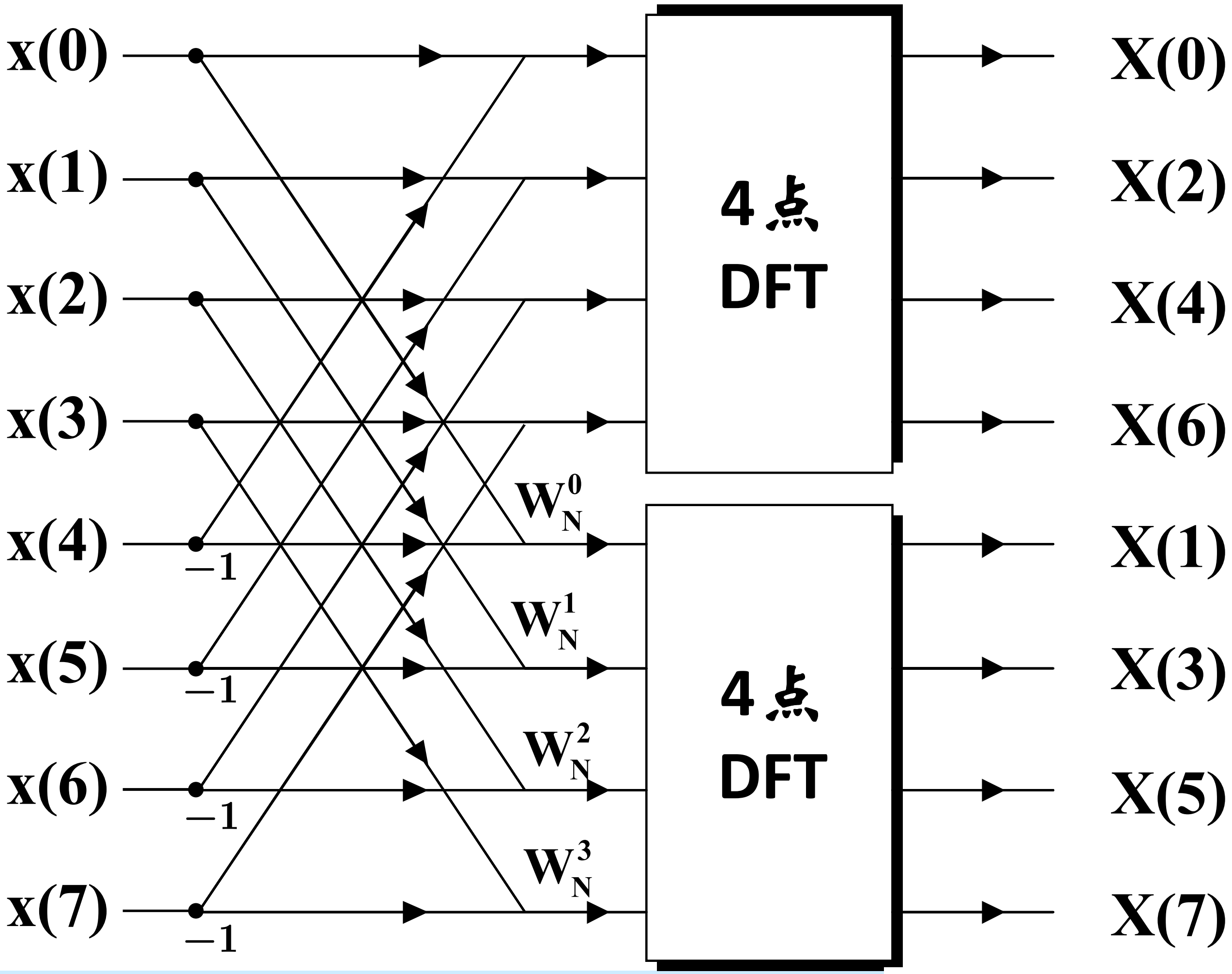
⟶ DIF-FFT算法

（DIT-FFT算法的分解类似）



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

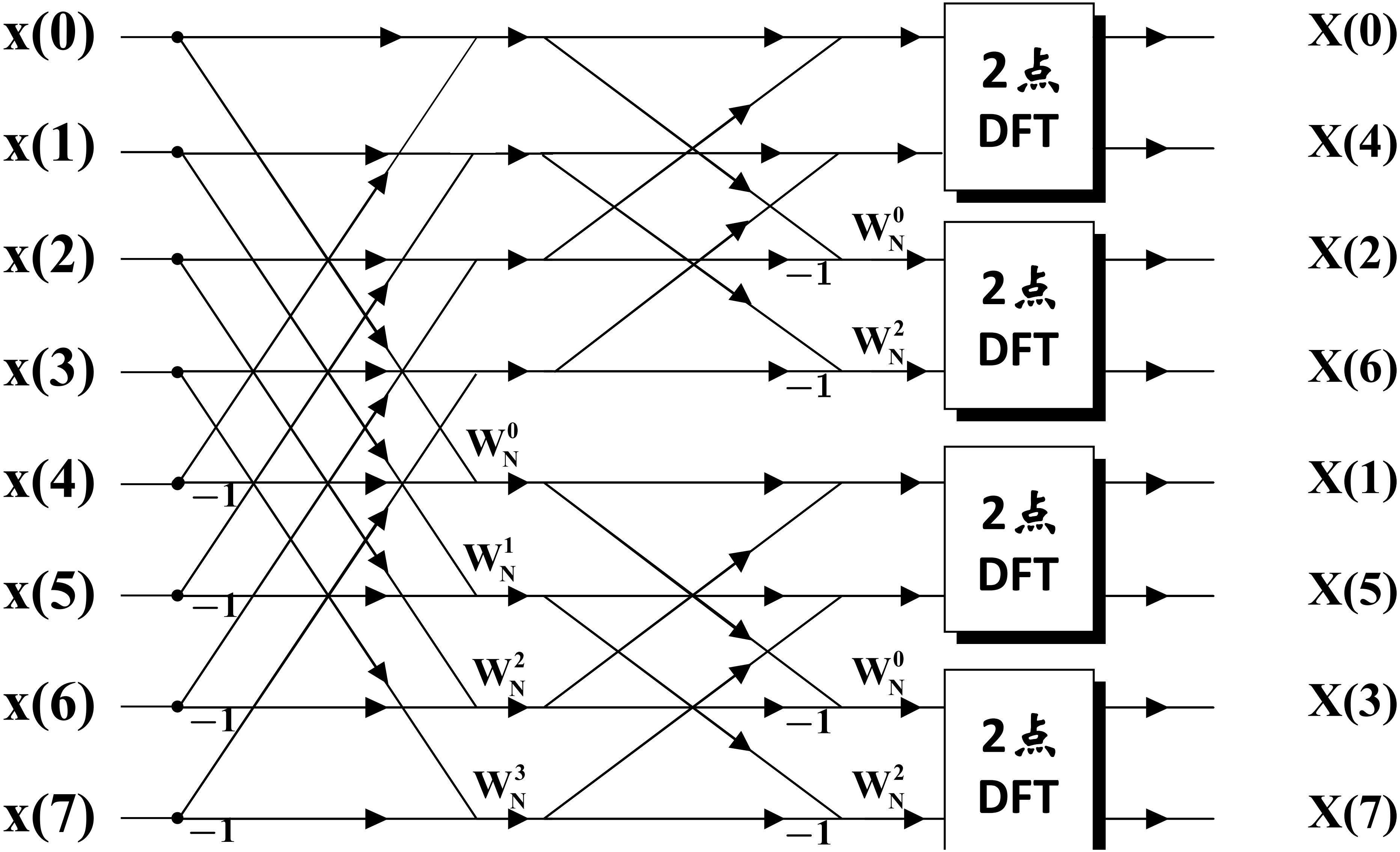
$$X(2k+1)=\sum_{n=0}^{N/2-1}[x(n)-x(n+N/2)]W_N^nW_{N/2}^{rk}\ ;k=0,1,\ldots,N/2-1$$




Cont'd ⇒

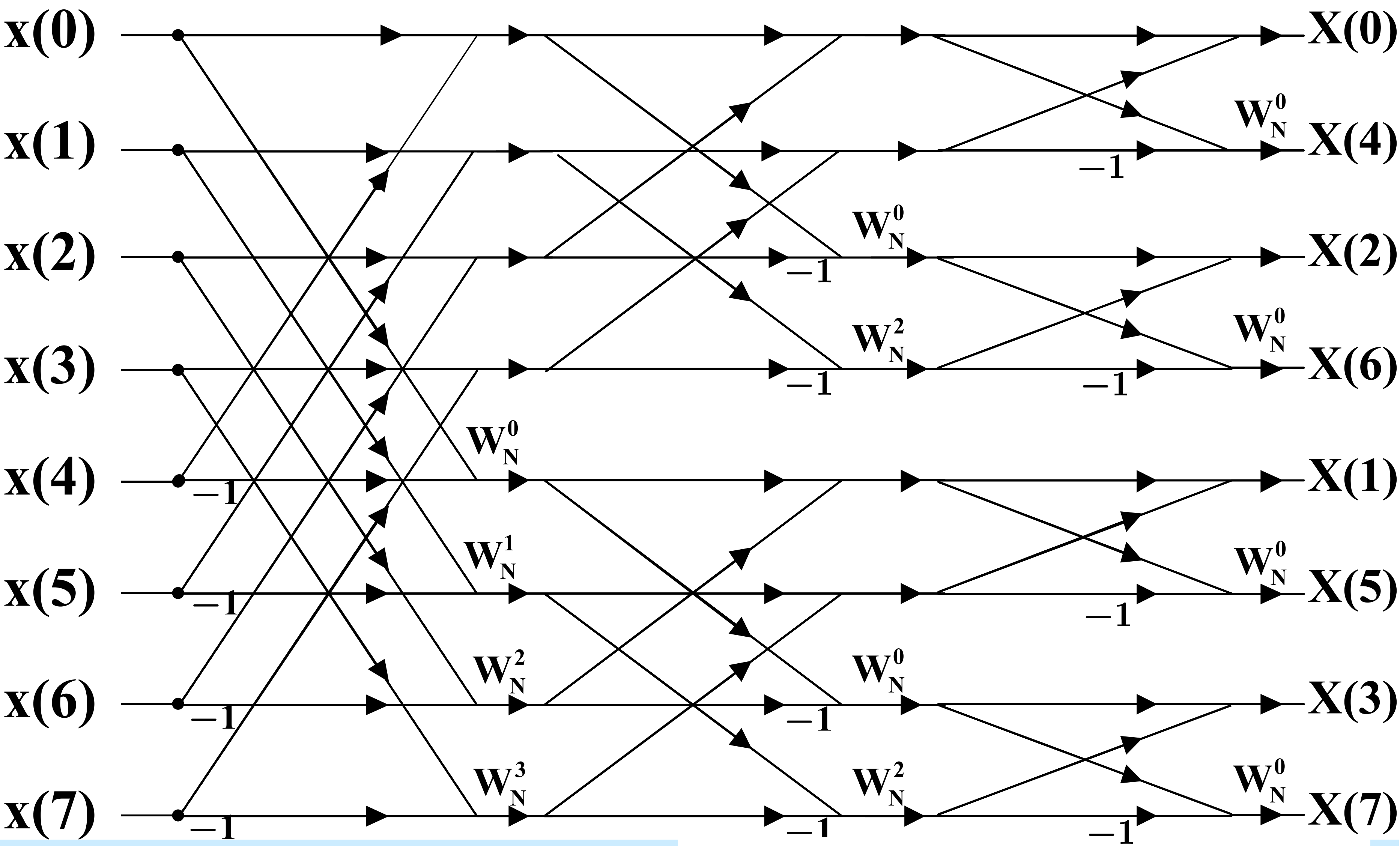


**8点基2频率抽取FFT算法流图



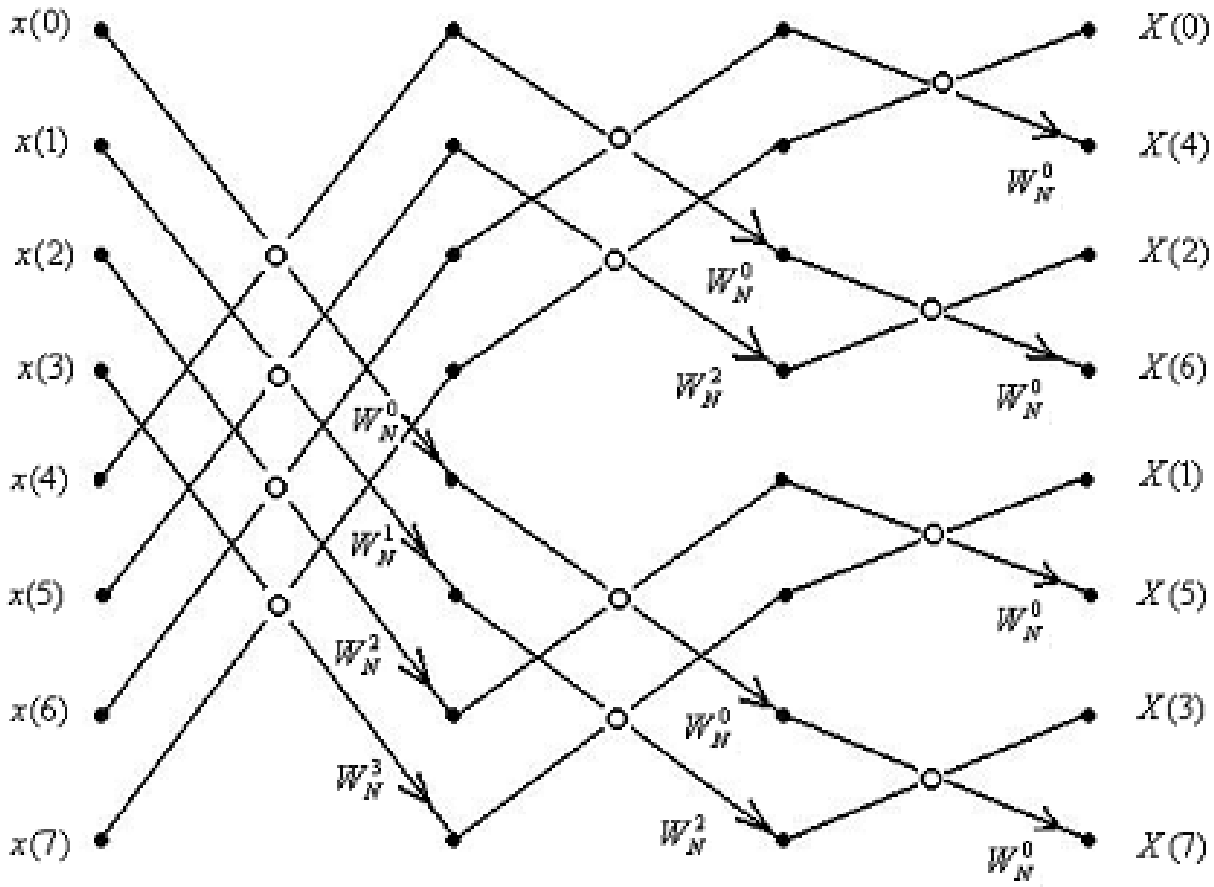

 Cont'd ⇒

**8点基2频率抽取FFT算法流图

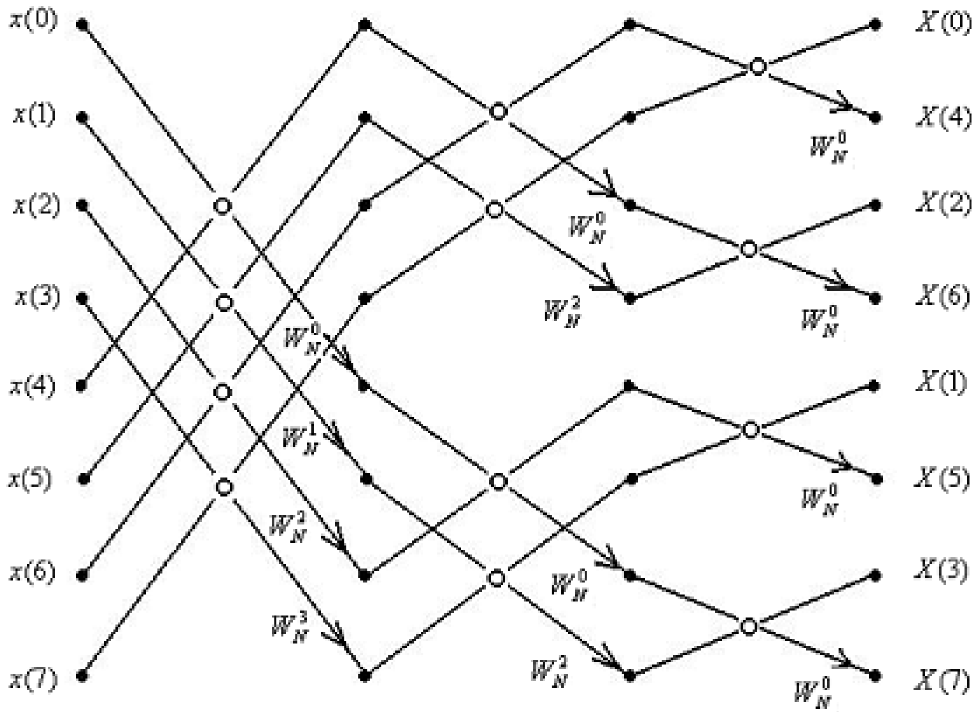



**8点基2频率抽取FFT算法流图

例：N=8，DIF-FFT算法流图

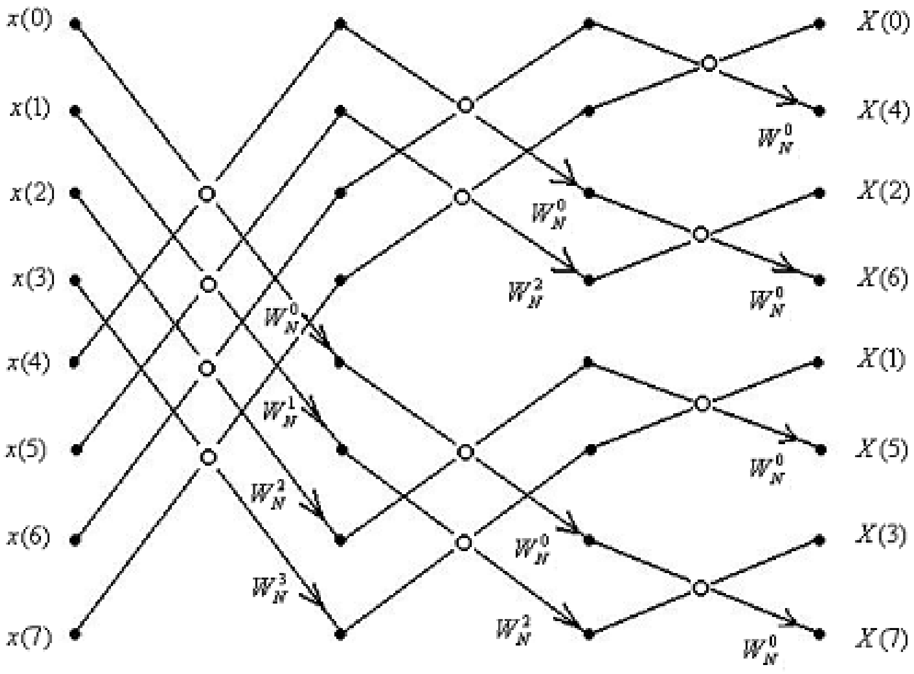


按频率抽取FFT规律

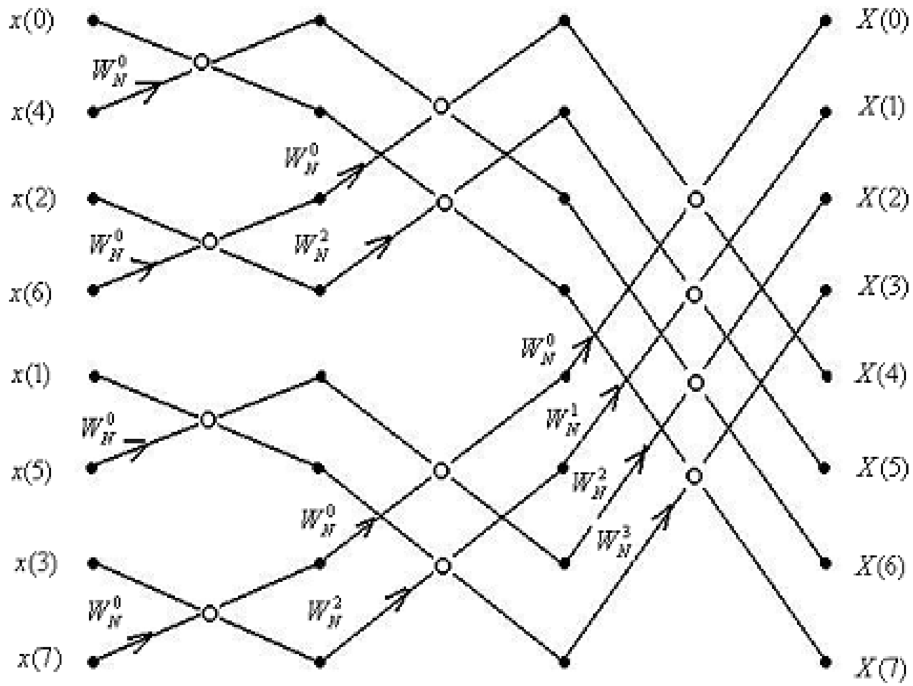



- 原位计算
- 距离
  - 列1: N/2 ------- $(2^{v-1})$
  - 列m: $2^{v-m}$
  - 列v: $2^0$
- $W_N^r$
  - 列1: $W_N^0,\quad W_N^1,\quad \ldots W_N^{\frac{N}{2}-1}$
  - 列2: $W_N^0,\quad W_N^2,\quad W_N^4\ldots$
  - 列v: $W_N^0$

## 二、DIF-FFT与DIT-FFT的比较




N=8,DIF-FFT




N=8,DIT-FFT

## 1. 二者的区别

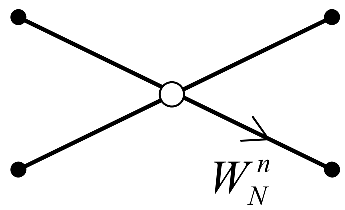
**(1) 输入与输出**（输入和输出的排序不是主要区别， 因为有变体形式）

**DIF**：顺序　　反序

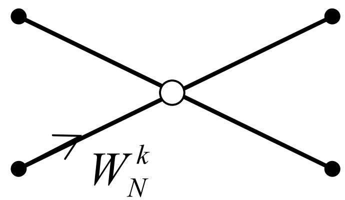
**DIT**：反序　　顺序

**(2) 蝶形运算** （关键区别）

**DIF**：先加减，后相乘




**DIT**：先相乘，后加减



## 2. 二者的相似之处

(1)分解过程

DIF： $v$ 列　　每列N/2个蝶形运算　　$m_F = \frac{N}{2}\log_2^N$， $a_F = \log_2^N$

DIT： $v$ 列　　同上　　同上

(2)原位运算（∵所有运算均由蝶形运算构成）

## 3. 二者关系

DIF-FFT　　转置　　DIT-FFT

## 三、逆DFT的快速算法(IFFT)

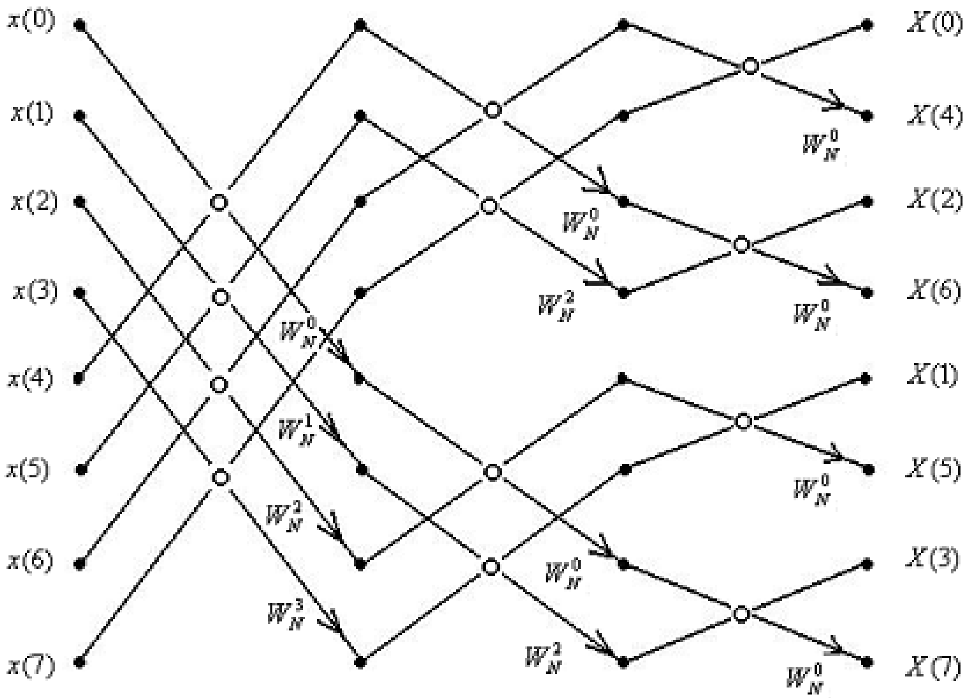
1. 算法一： $IDFT：\ x(n)=\frac{1}{N}\sum_{k=0}^{N-1}X(k)W_N^{-kn},\quad n=0,1,\dots,N-1$

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

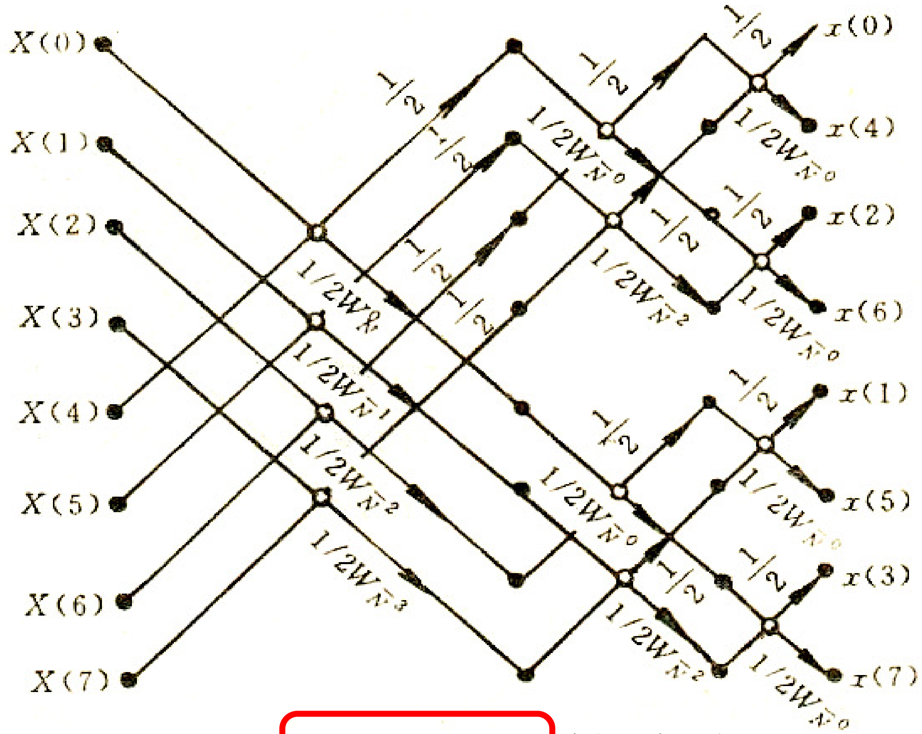
$$\text{FFT}\xrightarrow[x(n)\to X(k)]{W_N\to\frac{1}{2}W_N^{-1}}\text{IFFT}$$

(1) $$\text{DIT-FFT}\xrightarrow[x(n)\to X(k)]{W_N\to\frac{1}{2}W_N^{-1}}\text{DIF-IFFT}$$

(2) $$\text{DIF-FFT}\xrightarrow[x(n)\to X(k)]{W_N\to\frac{1}{2}W_N^{-1}}\text{DIT-IFFT}$$

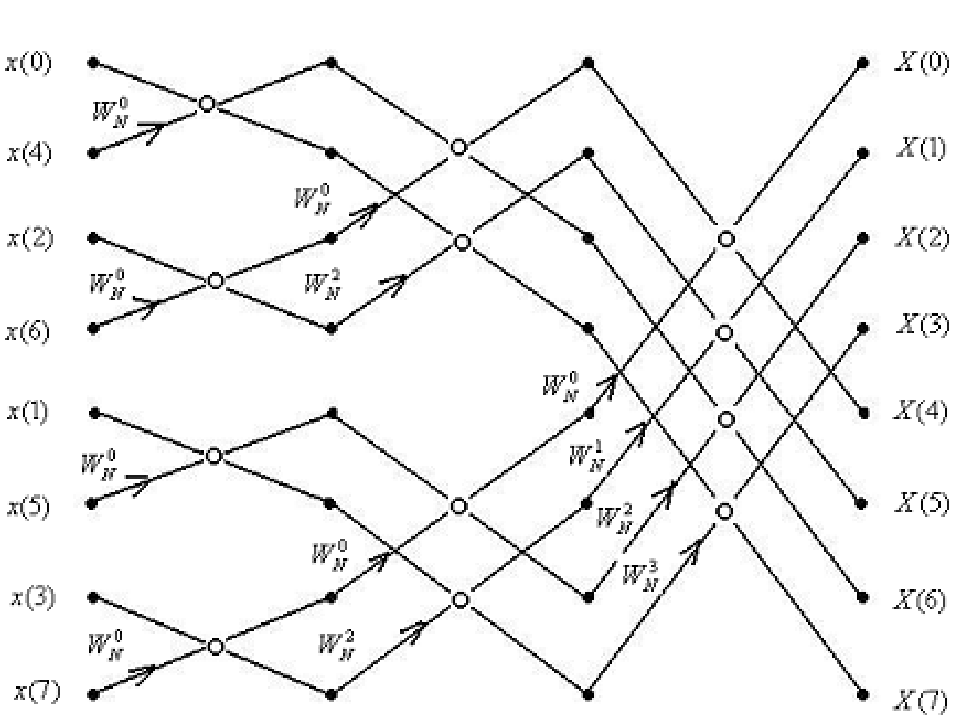



N=8，DIF-FFT算法流图

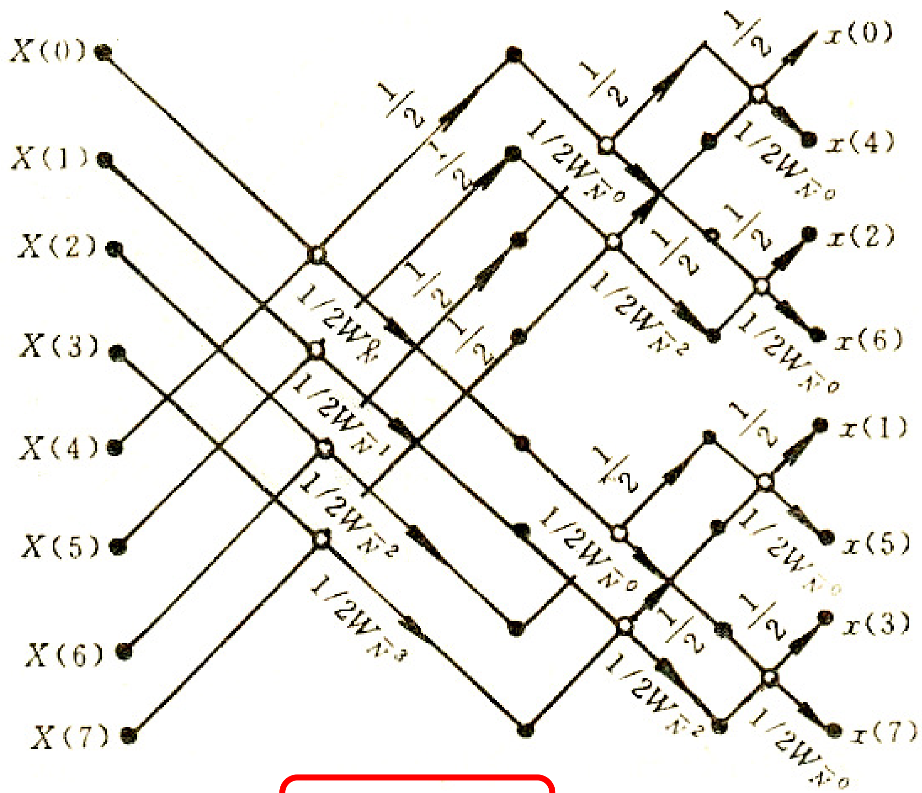



N=8，DIT-IFFT算法流图

解释：1.相同的流图可以采用同一个计算机算法
2.按时间抽取是“抽x(n)”和按频率抽取是“抽X(k)”
3.DIF-FFT和DIT-IFFT流图结构一样
4.DIT-FFT和DIF-IFFT流图结构一样




N=8，DIT-FFT运算流图




N=8，DIT-IFFT算法流图

2. 算法二

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

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

$$\downarrow$$

$$\frac{1}{N}\left\{FFT[X^*(k)]\right\}^*$$

**表 4-3　原位运算 FFT 的特点（$N=2^L$）**

| | 按时间抽选（DIT） | |
|---|---|---|
| | 输入自然数顺序、输出倒位序 | 输入倒位序、输出自然数顺序 |
| 蝶形结对偶节点距离 | $2^{L-m}=\frac{N}{2^m}$ | $2^{m-1}$ |
| 第 $m$ 级计算蝶形结计算公式 | $X_m(k)=X_{m-1}(k)+X_{m-1}\left(k+\frac{N}{2^m}\right)W_N^r$<br>$X_m\left(k+\frac{N}{2^m}\right)=X_{m-1}(k)-X_{m-1}\left(k+\frac{N}{2^m}\right)W_N^r$ | $X_m(k)=X_{m-1}(k)+X_{m-1}(k+2^{m-1})W_N^r$<br>$X_m(k+2^{m-1})=X_{m-1}(k)-X_{m-1}(k+2^{m-1})W_N^r$ |
| $W_N^r$ 中 $r$ 的求法 | 将地址 $k$ 除以 $2^{L-m}$（即右移（$L-m$）位）然后位序颠倒。具体步骤如下：<br>1. 把 $k$ 写成 $L$ 位二进制数；<br>2. 将此二进制数右移（$L-m$）位，把左边空出的位置补零；<br>3. 把已右移补零的二进制数位序颠倒，结果即为 $r$ 值。 | 将地址 $k$ 乘以 $2^{L-m}$（即左移（$L-m$）位）。具体步骤如下：<br>1. 把 $k$ 写成 $L$ 位二进制数；<br>2. 将此二进制数左移（$L-m$）位，把右边空出的位置补零，结果即为 $r$ 值。 |

| | 按频率抽选（DIF） | |
|---|---|---|
| | 输入自然顺序、输出倒位序 | 输入倒位序、输出自然顺序 |
| 蝶形结对偶节点距离 | $2^{L-m}=\frac{N}{2^m}$ | $2^{m-1}$ |
| 第 $m$ 级计算蝶形结计算公式 | $X_m(k)=X_{m-1}(k)+X_{m-1}\left(k+\frac{N}{2^m}\right)$<br>$X_m\left(k+\frac{N}{2^m}\right)=\left[X_{m-1}(k)-X_{m-1}\left(k+\frac{N}{2^m}\right)\right]W_N^r$ | $X_m(k)=X_{m-1}(k)+X_{m-1}(k+2^{m-1})$<br>$X_m(k+2^{m-1})=[X_{m-1}(k)-X_{m-1}(k+2^{m-1})]W_N^r$ |
| $W_N^r$ 中 $r$ 的求法 | 将地址 $k$ 乘以 $2^{m-1}$（即左移（$m-1$）位）。具体步骤如下：<br>1. 把 $k$ 写成 $L$ 位二进制数；<br>2. 将此二进制数左移（$m-1$）位，把右边空出的位置补零；结果即为 $r$ 值。 | 将地址 $k$ 除以 $2^{m-1}$（即右移（$m-1$）位），然后位序颠倒。具体步骤如下：<br>1. 把 $k$ 写成 $L$ 位二进制数；<br>2. 将此二进制数右移（$m-1$）位，把左边空出的位置补零；<br>3. 把已右移补零的二进制数位序颠倒，结果即为 $r$ 值。 |

# 作业

## 第四章

**1、4-5、8-13**