

## 第五章作业

题目：5.4、5.5、5.8、5.11、5.13、5.16

提交：时间待定

# 第 5 章 快速傅里叶变换FFT

郑南宁 教授

## 本章主要内容

- FFT算法的基本原理
- 基2FFT算法的理论推导
- 按时间抽取的FFT算法
- 按频率抽取的FFT算法
- IDFT的快速运算方法
- 利用FFT计算线性卷积和线性相关

## 5.1 FFT算法的基本原理

### ■ 矩阵方程 - DFT的计算量分析

$$X(k) = \sum_{n=0}^{N-1} x_0(n)W_N^{nk}, \quad k = 0, 1, \dots, N-1$$

上式表示  $N$  个方程的计算。为方便起见，下面用  $W^{nk}$  替代  $W_N^{nk}$  ( $W = e^{-j\frac{2\pi}{N}}$ )  
若  $N=4$ ，上式可展开为  $N$  个方程求解形式

$$\begin{aligned} X(0) &= x_0(0)W^0 + x_0(1)W^0 + x_0(2)W^0 + x_0(3)W^0 \\ X(1) &= x_0(0)W^0 + x_0(1)W^1 + x_0(2)W^2 + x_0(3)W^3 \\ X(2) &= x_0(0)W^0 + x_0(1)W^2 + x_0(2)W^4 + x_0(3)W^6 \\ X(3) &= x_0(0)W^0 + x_0(1)W^3 + x_0(2)W^6 + x_0(3)W^9 \end{aligned}$$

上面  $N$  个方程求解形式可表示成矩阵运算形式

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \quad (1-a)$$

把式(1 - a)重写如下

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \quad (1 - a)$$

或更紧凑地表示成

$$X(k) = W^{nk} x_0(n)$$

考察式(1 - a)，可以看到，由于 $W$ 是复数， $x_0(n)$ 也可能是复数，因此

- 1、每计算一个 $X(k)$ ，需要  $N$  次复数乘法， $N-1$  次复数加法
- 2、完成整个DFT运算，就需要  $N^2$  次复数乘法和  $N(N-1)$  次复数加法
- 3、每个复数乘法包含 4 次实数乘法和 2 次实数加法，每个复数加法包含 2 次实数加法

例如：当 $N=8$ 时，复乘次数=64，复加次数=56

当 $N=1024$ 时，复乘次数 $(1024)^2=1048576$

复加次数  $1024 \times (1024 - 1) = 1047552$

**DFT的复数相乘与复数相加的运算复杂度都为 $O(N^2)$ ，计算量大，难以做到实时处理**

## ■ 直观上讨论如何减少计算DFT所需的乘法和加法次数

把矩阵(1-a) 重写如下

$$\begin{aligned}
 \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} &= \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} & \xrightarrow{\text{(1-a) 改写为}} & \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} &= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 \\ 1 & W^2 & W^0 & W^2 \\ 1 & W^3 & W^2 & W^1 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} & \text{(1-b)}
 \end{aligned}$$

直观推导 (第一步) :

利用关系  $W^{nk} = W^{((nk))_N}$

因此, 如  $N = 4, n = 2, k = 3$  时, 则  $W^6 = W^2$

这是因为

$$\begin{aligned}
 W^{nk} &= W^6 = e^{(-j\frac{2\pi}{4})(6)} = e^{-j3\pi} \\
 &= W^{((nk))_N} = W^2 = e^{(-j\frac{2\pi}{4})(2)} = e^{-j\pi}
 \end{aligned}$$

直观推导 (第二步)

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 \\ 1 & W^2 & W^0 & W^2 \\ 1 & W^3 & W^2 & W^1 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \quad (1-c)$$

将式(1-c)中的矩阵 $[W^{nk}]$ 分解因子成如下形式

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \quad (1-d)$$

比较式(1-d)与式(1-c), 可以看到, 式(1-d)两个方阵相乘得到式(1-c)的方阵, 但其中第一行和第二行相互交换了。注意, 考虑这里的行交换, 需要改写式(1-d)的列矢量 $X(k)$ , 改写后的矢量用 $\overline{X(k)}$ 表示:

$$\overline{X(k)} = \begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} \quad (1-e)$$

式(1-c)矩阵因子分解是FFT算法之所以有效的关键 (注意其变换结果 $\overline{X(k)}$ 是乱序的)

重写式 (1-d)

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \quad (1-d)$$

承认上式的正确性 (尽管结果是乱序的), 就可以进一步讨论计算上述这个方程乘法次数

首先设 (先给出上式  
等号最右边两个矩阵  
相乘-中间计算步骤)

$$\begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \quad (1-f)$$

即, 列矢量  $x_1(n)$  等于式(1-d)右边两个矩阵的乘积, 元素  $x_1(0)$  的计算要用一次复数乘法和一次复数加法来确定, 即

$$x_1(0) = x_0(0) + W^0 x_0(2) \quad (1-g)$$

为了推得普遍的结果, 这里  $W^0$  不化为1

$x_1(1)$  的计算也是用一次复数乘法和一次复数加法来确定, 但计算  $x_1(2)$  只要一次复数加法, 这是因为,  $W^0 = -W^2$ , 因此

$$x_1(2) = x_0(0) + W^2 x_0(2) = x_0(0) - W^0 x_0(2) \quad (1-h)$$

上式中的复数乘法  $W^0 x_0(2)$  已在式 (1-g) 计算  $x_1(0)$  时计算过了

同理，计算 $x_1(3)$ 也只要一次复数加法，不需要乘法。这样中间矢量 $x_1(n)$ 只需要四次复数加法，不需作乘法。

下面继续完成式(1-d)的计算 (计算列矢量 $x_2(0)$ )

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \quad (1-d)$$

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} x_2(0) \\ x_2(1) \\ x_2(2) \\ x_2(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix} \quad (1-i)$$

$x_2(0)$ 项可用一次复数乘法和一次复数加法来确定，即

$$x_2(0) = x_1(0) + W^0 x_1(1) \quad (1-j)$$

$x_2(1)$ 的计算，只要一次复数加法，因为， $W^0 = -W^2$ 。同样， $x_2(2)$ 只要一次复数乘法和一次复数加法，而 $x_2(3)$ 的计算只要一次复数加法。

因此，用式(1-d)计算 $\overline{X(k)}$ ，总共需4次复数乘法和8次复数加法，而直接计算DFT需要16次复数乘法和12次复数加法

在前面计算式 (1-d) 时, 矩阵分解因子过程中, 由于把零引入了被分解的矩阵, 从而减少了乘法次数

- 对于  $N = 2^M$  的FFT算法, 简单地讲, 就是将  $N \times N$  的矩阵分解为  $M$  个矩阵分解因子 (每一个矩阵都具有复数加法和复数加法次数最少的特性)
- 引伸上例的结果, 可以看到,  $N=2^M$  的FFT算法只需要  $N \cdot M/2$  次复数乘法和  $N \cdot M$  次复数加法, 而直接计算DFT需要  $N^2$  次复数乘法和  $N(N-1)$  次复数加法

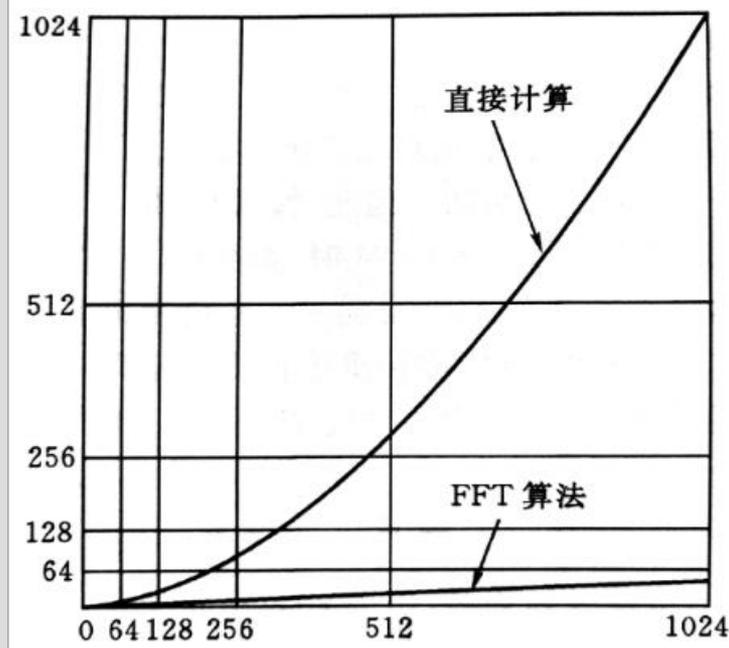
若只考虑乘法次数, 直接计算DFT与FFT的计算时间之比的近似关系

$$\frac{N^2}{N \cdot M/2} = \frac{2N}{M}$$

对于  $N=1024=2^{10}$ , 可以使计算量减少到约200比1

计算复杂度从  $O(N^2)$  下降为  $O(N \log(N))$

- **FFT算法的基本原理:** 将大  $N$  点数的DFT分解为若干小点数的DFT的组合, 使整个DFT的计算过程变成一系列迭代运算过程



**N点DFT和FFT所需乘法次数的比较**

在前面计算式(1-d)时,矩阵分解因子过程中引进了一个差异,在计算式(1-d)时,得到的是 $\overline{X(k)}$ ,而不是 $X(k)$ ,即

$$\overline{X(k)} = \begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} \quad \text{代替了} \quad X(k) = \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} \quad (1-k)$$

这种重新排列是矩阵分解过程固有的。可直接对 $\overline{X(k)}$ 重新排序得到 $X(k)$

### 对结果重新排序

用相应的二进制数代替自变量 $k$ , 重写 $X(k)$

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} \quad \text{变成} \quad \begin{bmatrix} X(00) \\ X(10) \\ X(01) \\ X(11) \end{bmatrix}$$

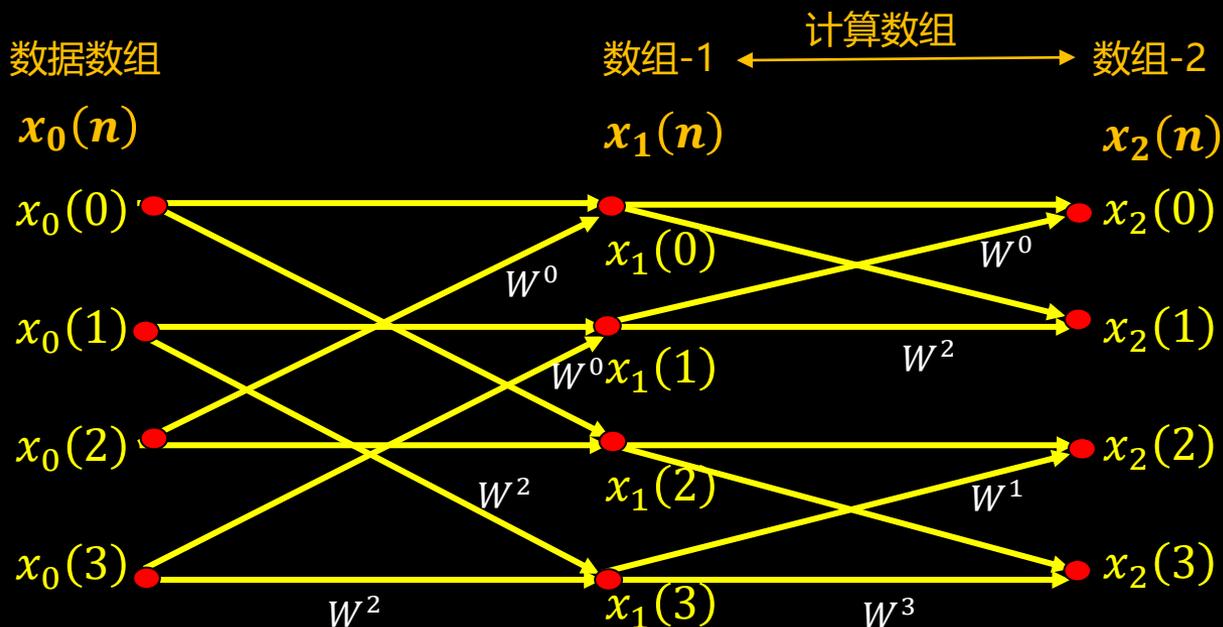
将上面矩阵的二进制数位序反转

$$\overline{X(k)} = \begin{bmatrix} X(00) \\ X(10) \\ X(01) \\ X(11) \end{bmatrix} \quad \text{反转} \quad \begin{bmatrix} X(00) \\ X(01) \\ X(10) \\ X(11) \end{bmatrix} = X(k)$$

以上讨论了 $N=4$ 的FFT算法,对于 $N > 4$ 的FFT算法按式(1-d)的方式描述矩阵因子分解过程比较麻烦,后面将用图解的方式讨论式(1-d)的计算过程,用这种形式导出计算机的流程图

## ■ 信号流程图 (计算式(1-d)的信号流程图)

左边第一列是数据矢量或数组 $x_0(n)$ ，第二列结点是式(1-f)所计算的矢量 $x_1(n)$ ，第三列结点对应于 $x_2(n) = \overline{X(k)}$  (式(1-i))。如果 $N = 2^M$ ，则有 $M$ 个计算数组



**信号流程：**线条表示传输路径，进入每一个结点的两条线表示来自前一列的两个结点的传输或量值，这个量值乘以系数 $W^p$ ，然后把相乘的结果输入到这一列的结点；当没有这个系数，表示 $W^p=1$ ，两条传输路径进到一个结点的结果要相加

## ■ 减少DFT计算量的依据

改进DFT计算效率的大多数方法利用了 $W_N^{nk}$ 的周期性和对称性

(1)  $W_N^{nk}$ 对  $n$  和  $k$  的周期性  $W_N^{(n+N)k} = W_N^{n(k+N)} = W_N^{nk}$

(2)  $W_N^{nk}$ 的复共轭对称性  $(W_N^{nk})^* = W_N^{k(N-n)} = W_N^{-nk}$

另外, 由于 $W_N^{N/2} = e^{-j\pi} = -1$ , 则有

$$W_N^{(k+N/2)} = -W_N^k$$

同时利用 $W_N^{nk}$ 的周期性和对称性可以大大减少DFT的计算量

**举例** 对实数序列 $x(n)$ , 利用权重的对称性, 则可将其DFT公式中含有 $n$ 和 $N-n$ 的项组合在一起

$$\begin{aligned}
 & \underline{x(n)W_N^{nk} + x(N-n)W_N^{(N-n)k}} \\
 &= x(n)W_N^{nk} + x(N-n)(W_N^{nk})^* \\
 &= \underline{[x(n) + x(N-n)]\text{Re}(W_N^{nk}) + j[x(n) - x(N-n)]\text{Im}(W_N^{nk})}
 \end{aligned}$$

原式需 4 次实数乘法

变形以后需 2 次实数乘法

对其它项也做类似的组合处理, 计算DFT的乘法次数就减少一半

## ■ 基2FFT算法的理论推导 ( $N=4$ 时的理论证明)

### (1) 记号的定义

考虑DFT变换式 ( $N = 2^M$ )

$$X(k) = \sum_{n=0}^{N-1} x_0(n) W_N^{nk}, \quad k = 0, 1, \dots, N-1 \quad (2-a)$$

将整数  $n$  和  $k$  用二进制数表示; 假设  $N = 4$ , 那么  $M=2$ ,  $n$  和  $k$  的二进制数表示为

$$n = 0, 1, 2, 3 \quad \text{对应} \quad n = (n_1, n_0) = 00, 01, 10, 11$$

$$k = 0, 1, 2, 3 \quad \text{对应} \quad k = (k_1, k_0) = 00, 01, 10, 11$$

把  $k$  和  $n$  写成如下紧凑形式:

$$n = 2n_1 + n_0, \quad k = 2k_1 + k_0 \quad (2-b)$$

这里  $n_0, n_1, k_0$  和  $k_1$  只能取值 0 和 1。上式是把相应的十进制数写成二进制数的一种方法

当  $N = 4$  时，可把式(2-a)改写为（为方便起见，用  $W^{nk}$  替代  $W_N^{nk}$ ）

$$X(k_1, k_0) = \sum_{n_0=0}^1 \sum_{n_1=0}^1 x_0(n_1, n_0) W^{(2k_1+k_0)(2n_1+n_0)} \quad (2-c)$$

注意，为了计算二进制表示的  $n$  的所有位，公式(2-a)中单个求和号现在必须用  $M$  个求和号来代替

## (2) $W^p$ 的因子分解

现在研究  $W^p$  这一项。由于  $W^{a+b} = W^a \cdot W^b$ ，所以

$$\begin{aligned} W^{(2k_1+k_0)(2n_1+n_0)} &= W^{(2k_1+k_0)2n_1} \cdot W^{n_0(2k_1+k_0)} \\ &= [W^{4k_1n_1}] W^{2k_0n_1} W^{n_0(2k_1+k_0)} = W^{2k_0n_1} W^{(2k_1+k_0)n_0} \end{aligned} \quad (2-d)$$

注意，方括号中的项等于1，因为

$$\underline{W^{4k_1n_1}} = [W^4]^{k_1n_1} = [e^{-j2\pi 4/4}]^{k_1n_1} = [1]^{k_1n_1} = 1 \quad (2-e)$$

这样，式(2-c)就可以写成下面的形式

$$X(k_1, k_0) = \sum_{n_0=0}^1 \left[ \sum_{n_1=0}^1 x_0(n_1, n_0) W^{2n_1 k_0} \right] W^{n_0(2k_1+k_0)} \quad (2-f)$$

上式代表了FFT算法的基础。为了说明这一点，下面分别研究上式中各个求和式

首先讨论上式方括号内的求和式

$$x_1(k_0, n_0) = \sum_{n_1=0}^1 x_0(n_1, n_0) W^{2n_1 k_0} \quad (2-g)$$

将上式展开，有

$$\begin{cases} x_1(0,0) = x_0(0,0) + x_0(1,0)W^0 \\ x_1(0,1) = x_0(0,1) + x_0(1,1)W^0 \\ x_1(1,0) = x_0(0,0) + x_0(1,0)W^2 \\ x_1(1,1) = x_0(0,1) + x_0(1,1)W^2 \end{cases} \quad (2-h)$$

把式(2-h)重写

$$\begin{cases} x_1(0,0) = x_0(0,0) + x_0(1,0)W^0 \\ x_1(0,1) = x_0(0,1) + x_0(1,1)W^0 \\ x_1(1,0) = x_0(0,0) + x_0(1,0)W^2 \\ x_1(1,1) = x_0(0,1) + x_0(1,1)W^2 \end{cases} \quad (2-h)$$

如果把式(2-h)表示成矩阵形式, 有

$$\begin{bmatrix} x_1(0,0) \\ x_1(0,1) \\ x_1(1,0) \\ x_1(1,1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0,0) \\ x_0(0,1) \\ x_0(1,0) \\ x_0(1,1) \end{bmatrix} \quad (2-i)$$

注意, 上式正好是前面第7页PPT中讨论的矩阵方程式 (1-f)

$$\begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \quad (1-f)$$

但式(2-i)中的 $n$ 是用二进制表示的

因此，式(2-f)方括号

$$X(k_1, k_0) = \sum_{n_0=0}^1 \left[ \sum_{n_1=0}^1 x_0(n_1, n_0) W^{2k_0 n_1} \right] W^{(2k_1+k_0)n_0} \quad (2-f)$$

里边的求和式代表5.1节直观推导DFT计算次数时的式(1-d)的第一个矩阵因子

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \quad (1-d)$$

同样，如果把式(2-f)

$$X(k_1, k_0) = \sum_{n_0=0}^1 \left[ \sum_{n_1=0}^1 x_0(n_1, n_0) W^{2n_1 k_0} \right] W^{n_0(2k_1+k_0)} \quad (2-f)$$

的外层的求和写成

$$x_2(k_0, k_1) = \sum_{n_0=0}^1 x_1(k_0, n_0) W^{(2k_1+k_0)n_0} \quad (2-j)$$

进一步写成矩阵形式，得到

$$\begin{bmatrix} x_2(0,0) \\ x_2(0,1) \\ x_2(1,0) \\ x_2(1,1) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} x_1(0,0) \\ x_1(0,1) \\ x_1(1,0) \\ x_1(1,1) \end{bmatrix} \quad (2-k)$$

这就是矩阵方程(1-f)。所以式(2-f)外层的求和定义了前面第6页PPT直观推导DFT计算次数时的式(1-d)的第二个矩阵因子

$$\begin{bmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{bmatrix} \begin{bmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \end{bmatrix} \quad (1-d)$$

把公式(2-f)和(2-j) 重写如下

$$X(k_1, k_0) = \sum_{n_0=0}^1 \left[ \sum_{n_1=0}^1 x_0(n_1, n_0) W^{2n_1 k_0} \right] W^{n_0(2k_1+k_0)} \quad (2-f)$$

$$x_2(k_0, k_1) = \sum_{n_0=0}^1 x_1(k_0, n_0) W^{(2k_1+k_0)n_0} \quad (2-j)$$

于是, 有

$$X(k_1, k_0) = x_2(k_0, k_1) \quad (2-l)$$

也就是说, 由外层求和得到的最后结果 $x_2(k_0, k_1)$ 与我们所要求的 $X(k_1, k_0)$ , 它们的位序是倒序的

如果把公式(2-g)、(2-j)和(2-l)相联立, 即

$$\begin{cases} x_1(k_0, n_0) = \sum_{n_1=0}^1 x_0(n_1, n_0) W^{2n_1 k_0} \\ x_2(k_0, k_1) = \sum_{n_0=0}^1 x_1(k_0, n_0) W^{n_0(2k_1+k_0)} \\ X(k_1, k_0) = x_2(k_0, k_1) \end{cases} \quad (2-m)$$

上面方程组(2-m)便是库利-图基最初为 $N = 4$ 列出的FFT算法。在这一组方程中, 第二个方程是由第一个方程计算的, 所以称它们是递归的

## 5.2 按时间抽取的FFT算法

### ■ 算法原理 (基2-FFT)

先将 $x(n)$ 按 $n$ 的奇偶分为两组, 设 $N=2^M$ , 不足时补零,  $n$ 用变量 $2r$ 表示, 即

$$n \text{ 为偶数: } x(2r) = x_1(r), \quad r = 0, 1, \dots, \frac{N}{2} - 1$$

$$n \text{ 为奇数: } x(2r + 1) = x_2(r), \quad r = 0, 1, \dots, \frac{N}{2} - 1$$

因此有

$$\begin{aligned} X(k) &= \text{DFT}[x(n)] = \sum_{n=0}^{N-1} x(n)W_N^{nk} \\ &= \sum_{n=0}^{N-1} x(n)W_N^{nk} + \sum_{n=0}^{N-1} x(n)W_N^{nk} \\ &= \sum_{r=0}^{\frac{N}{2}-1} x(2r)W_N^{2rk} + \sum_{r=0}^{\frac{N}{2}-1} x(2r+1)W_N^{(2r+1)k} \\ &= \sum_{r=0}^{\frac{N}{2}-1} x_1(r)(W_N^2)^{rk} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x_2(r)(W_N^2)^{rk} \end{aligned}$$

由于  $W_N^2 = e^{-j\frac{2\pi}{N}2} = e^{-j2\pi/(\frac{N}{2})} = W_{N/2}$ , 上式可表示为:

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} x_1(r) W_{\frac{N}{2}}^{rk} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x_2(r) W_{\frac{N}{2}}^{rk} = X_1(k) + W_N^k X_2(k)$$

式中  $k = 0, \dots, \frac{N}{2} - 1$

其中

$$\left\{ \begin{array}{l} X_1(k) = \sum_{r=0}^{\frac{N}{2}-1} x_1(r) W_{\frac{N}{2}}^{rk} = \text{DFT}(x_1(r)) \\ X_2(k) = \sum_{r=0}^{\frac{N}{2}-1} x_2(r) W_{\frac{N}{2}}^{rk} = \text{DFT}(x_2(r)) \end{array} \right.$$

## ■ 结论

- 1、 $X_1(k)$ ,  $X_2(k)$ 均为 $N/2$ 点的DFT
- 2、 $X(k) = X_1(k) + W_N^k X_2(k)$ 只能确定出 $X(k)$ 的 $k=0, 1, \dots, \frac{N}{2} - 1$ 的值, 即 $X(k)$ 前一半的结果

## ■ $X(k)$ 后一半的计算

根据 $W_N^{nk}$ 的周期性有 $W_{N/2}^{r(k+N/2)} = W_{N/2}^{rk}$ , 所以有:

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

同理有  $X_2\left(\frac{N}{2} + k\right) = X_2(k) \quad k = 0, \dots, \frac{N}{2} - 1$

这就是说,  $X_1(k)$ 、 $X_2(k)$ 的后一半, 分别等于其前一半的值

又由于 $W_N^{(N/2+k)} = W_N^{N/2} W_N^k = -W_N^k$ , 且 $X_1(k)$ 和 $X_2(k)$ 以 $N/2$ 为周期, 故有:

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

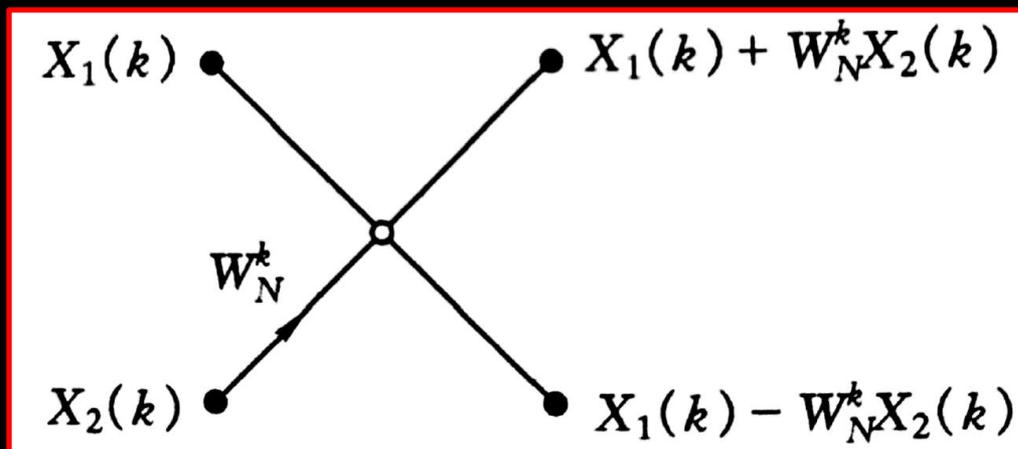
可见,  $X(k)$ 的后一半也完全由 $X_1(k)$ 、 $X_2(k)$ 所确定。所以,  $N$ 点的DFT可由两个 $N/2$ 点的DFT来计算

## ■ 蝶形运算

由 $X_1(k)$ 、 $X_2(k)$ 来表示 $X(k)$ 的运算如下

$$\begin{cases} X(k) = X_1(k) + W_N^k X_2(k) \\ X(k + \frac{N}{2}) = X_1(k) - W_N^k X_2(k) \end{cases} \quad (k = 0, 1, \dots, \frac{N}{2} - 1)$$

实现上式运算的流图称作**蝶形运算**



## ■ 举例

$N=8$  序列的DFT, 可以分解为两个 $N/2=4$ 点的DFT, 具体方法如下:

1、 $n$  为偶数时的序列  $x_1(n)$  为

$$x_1(0) = x(0), x_1(1) = x(2), x_1(2) = x(4), x_1(3) = x(6)$$

进行  $N/2=4$  点的DFT得  $X_1(k)$

$$X_1(k) = \sum_{r=0}^3 x_1(r)W_4^{rk} = \sum_{r=0}^3 x(2r)W_4^{rk}, k = 0,1,2,3$$

2、 $n$  为奇数时的序列  $x_2(n)$  为

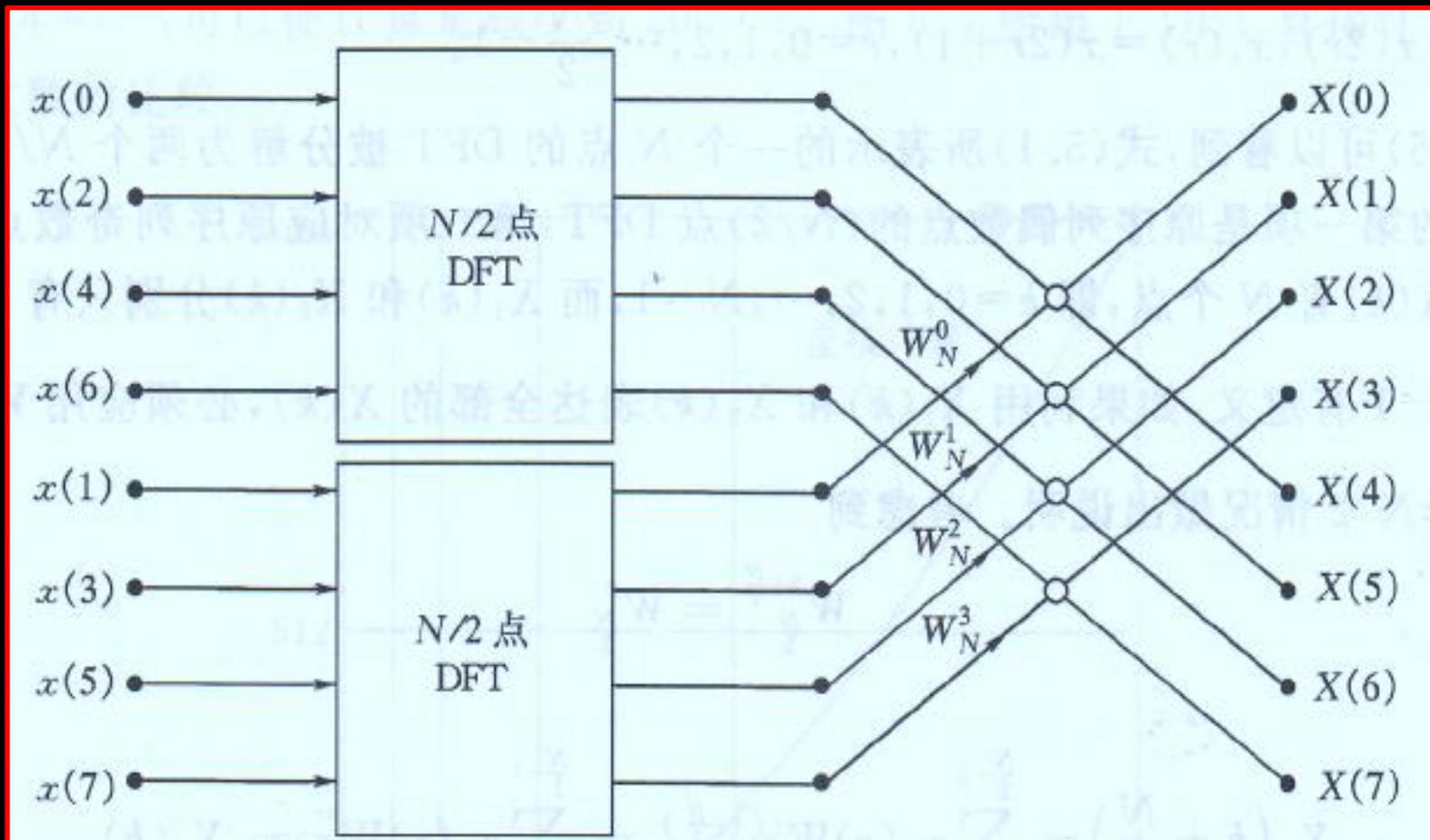
$$x_2(0) = x(1), x_2(1) = x(3), x_2(2) = x(5), x_2(3) = x(7)$$

进行 $N/2=4$ 点的DFT得  $X_2(k)$

$$X_2(k) = \sum_{r=0}^3 x_2(r)W_4^{rk} = \sum_{r=0}^3 x(2r+1)W_4^{rk}, k = 0,1,2,3$$

根据蝶形运算, 即可由4点 $X_1(k)$ 和4点 $X_2(k)$ 来计算全部 $N=8$ 点  $X(k)$

3、对 $X_1(k)$ 和 $X_2(k)$ 进行蝶形运算，前半部为 $X(0)$ 到 $X(3)$ ，后半部分为 $X(4)$ 到 $X(7)$ ，整个过程如下图所示：



## ■ 迭代奇偶分组

按照奇偶分组的基本思想，对每个 $N/2$ 的序列进一步奇偶分组，从而进一步减少运算量。假设序列总长 $N=2^L$ ，就可以这样分解 $L$ 层。当不满足时，可以补零操作

1、对前面的偶序列 $x_1(r)$ ,  $r = 0, \dots, \frac{N}{2} - 1$ 进一步奇偶数分解，即

$$x_1(2l) = x_3(l), \quad l = 0, \dots, \frac{N}{4} - 1$$

$$x_1(2l+1) = x_4(l), \quad l = 0, \dots, \frac{N}{4} - 1$$

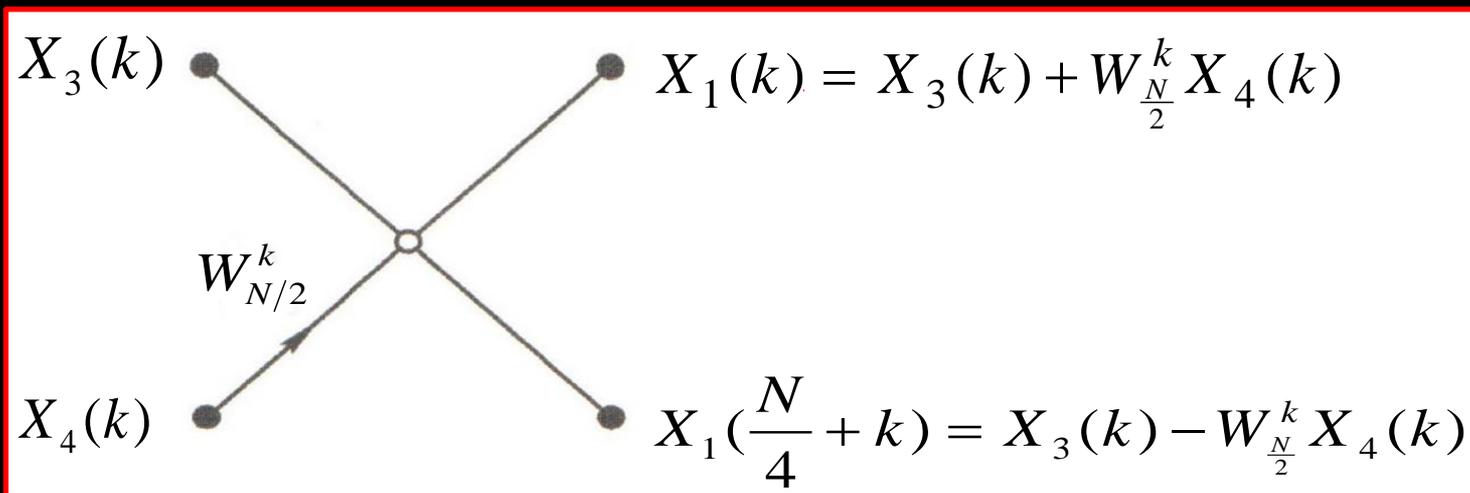
分别进行 $N/4$ 点的DFT，得到：

$$\begin{cases} X_3(k) = \sum_{l=0}^{\frac{N}{4}-1} x_3(l)W_{N/4}^{lk}, & k = 0, \dots, \frac{N}{4} - 1 & \text{(偶数序列中再取偶)} \\ X_4(k) = \sum_{l=0}^{\frac{N}{4}-1} x_4(l)W_{N/4}^{lk}, & k = 0, \dots, \frac{N}{4} - 1 & \text{(偶数序列中再取奇)} \end{cases}$$

利用蝶形运算, 可用 $N/4$ 点的 $X_3(k)$ 和 $X_4(k)$ 计算 $N/2$ 点的 $X_1(k)$ , 即

$$\begin{cases} X_1(k) = X_3(k) + W_{\frac{N}{2}}^k X_4(k) \\ X_1(\frac{N}{4} + k) = X_3(k) - W_{\frac{N}{2}}^k X_4(k) \end{cases} \quad k = 0, 1, \dots, \frac{N}{4} - 1$$

其蝶形运算流程图如下



2、同样对前面的奇序列  $x_2(r)$ ,  $r = 0, \dots, \frac{N}{2} - 1$  进一步奇偶数分解为  $N/4$  的序列

$x_5(l)$  和  $x_6(l)$ , 并分别进行  $N/4$  点的DFT如下

$$\begin{cases} X_5(k) = \sum_{l=0}^{N/4-1} x_2(2l)W_{N/4}^{lk} = \sum_{l=0}^{N/4-1} x_5(l)W_{N/4}^{lk} & \text{(奇数序列中再取偶序号子序列)} \\ X_6(k) = \sum_{l=0}^{N/4-1} x_2(2l+1)W_{N/4}^{lk} = \sum_{l=0}^{N/4-1} x_6(l)W_{N/4}^{lk} & \text{(奇数序列中再取奇序号子序列)} \end{cases}$$

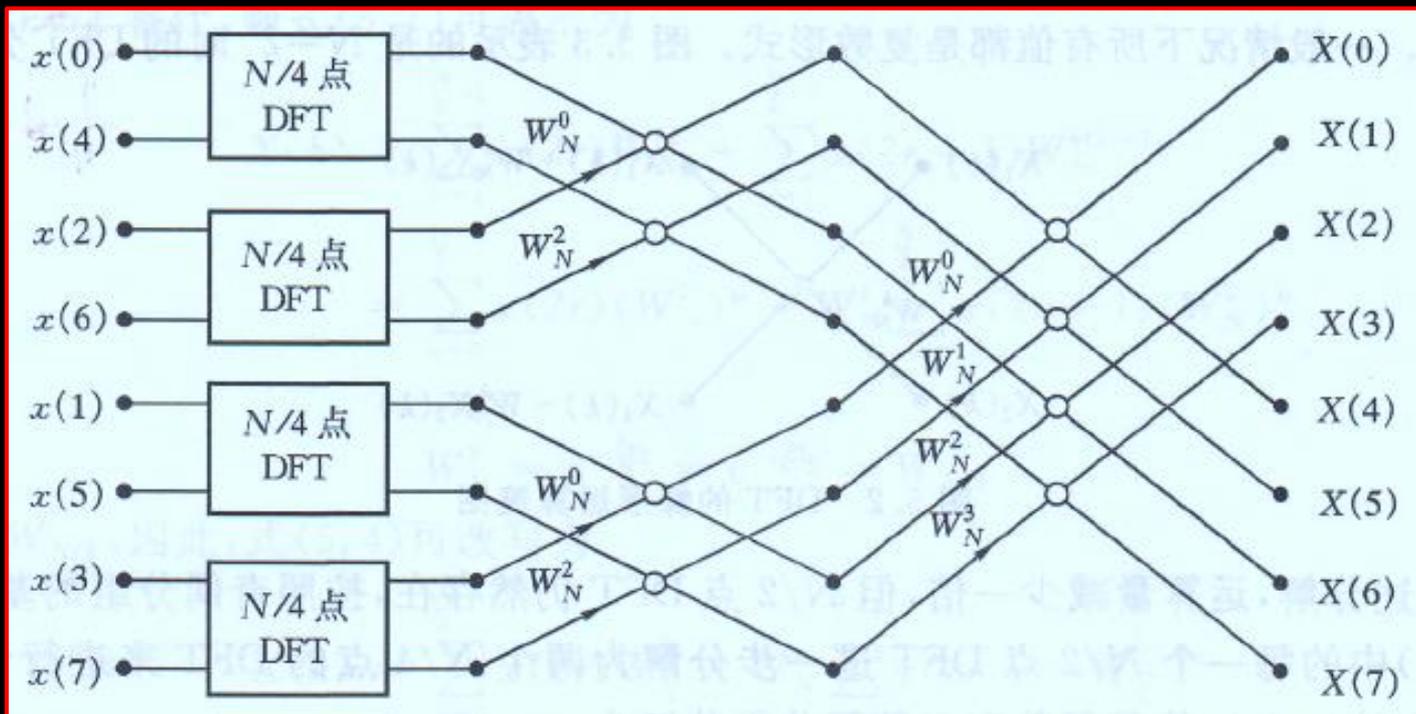
由  $X_5(k)$ 、 $X_6(k)$  进行蝶形运算计算  $N/2$  点的  $X_2(k)$ , 得到

$$\begin{cases} X_2(k) = X_5(k) + W_{N/2}^k X_6(k); k = 0, 1, \dots, \frac{N}{4} - 1 \\ X_2\left(\frac{N}{4} + k\right) = X_5(k) - W_{N/2}^k X_6(k); k = 0, 1, \dots, \frac{N}{4} - 1 \end{cases}$$

所以,  $N$  点的DFT分解为四个  $N/4$  点的DFT来计算

### 3、两级蝶形计算流程图如下

(为减少参数个数, 设  $W_{N/2}^0 = W_N^0$ ,  $W_{N/2}^1 = W_N^2$ )



经过上述两级蝶形运算, 用四个  $N/4$  点 DFT 计算  $N$  点序列的 DFT, 其运算量可再减少约一半, 即为  $N$  点 DFT 计算量的  $1/4$

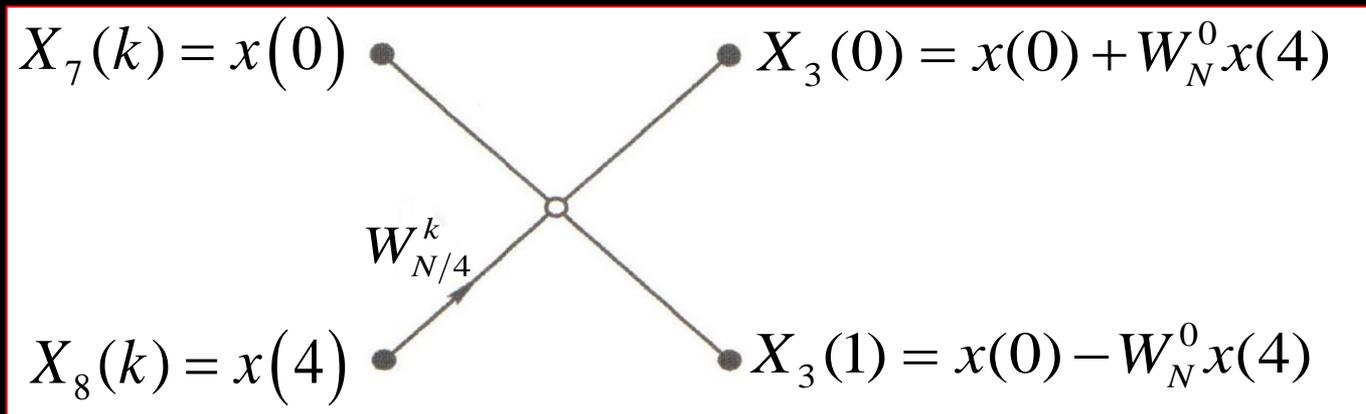
#### 4、最后一级——2点DFT的蝶形运算

对于  $N=2^3=8$  时DFT，经过 2 级迭代，即为  $N/4$  点即为两点DFT。

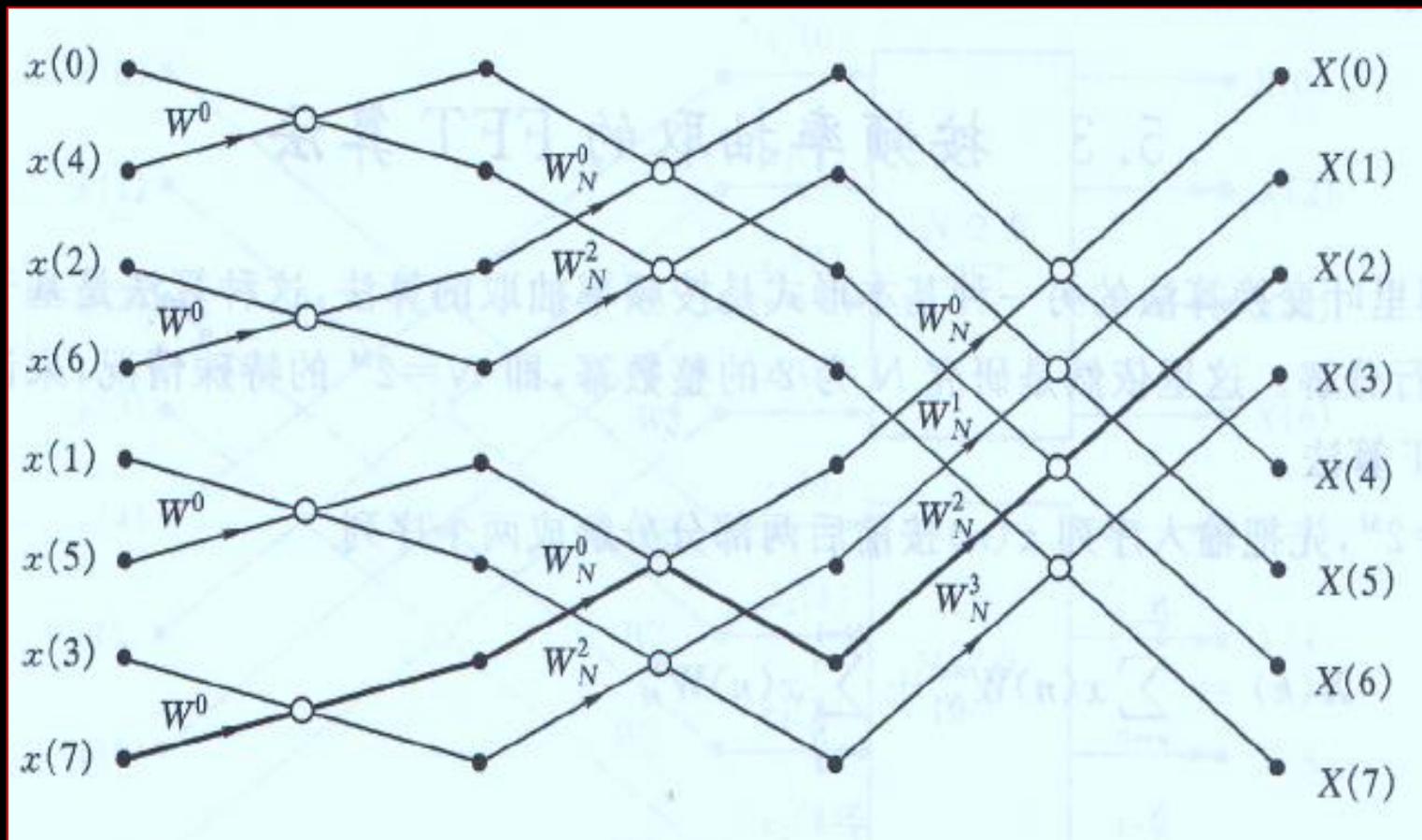
以  $X_3(k) = \sum_{l=0}^1 x_3(l)W_2^{lk}$ ,  $k=0,1$  为例，即

$$\begin{cases} X_3(0) = x_7(0) + W_{N/4}^0 x_8(0) = x(0) + W_N^0 x(4) \\ X_3(1) = x_7(0) - W_{N/4}^0 x_8(0) = x(0) - W_N^0 x(4) \end{cases}$$

2点DFT仍可以用蝶形运算分解为单点DFT，进一步减少运算量，即



5、最终，得到  $N=2^M=8$  点DFT分解为  $M=3$  级迭代的蝶形运算流程图如下：

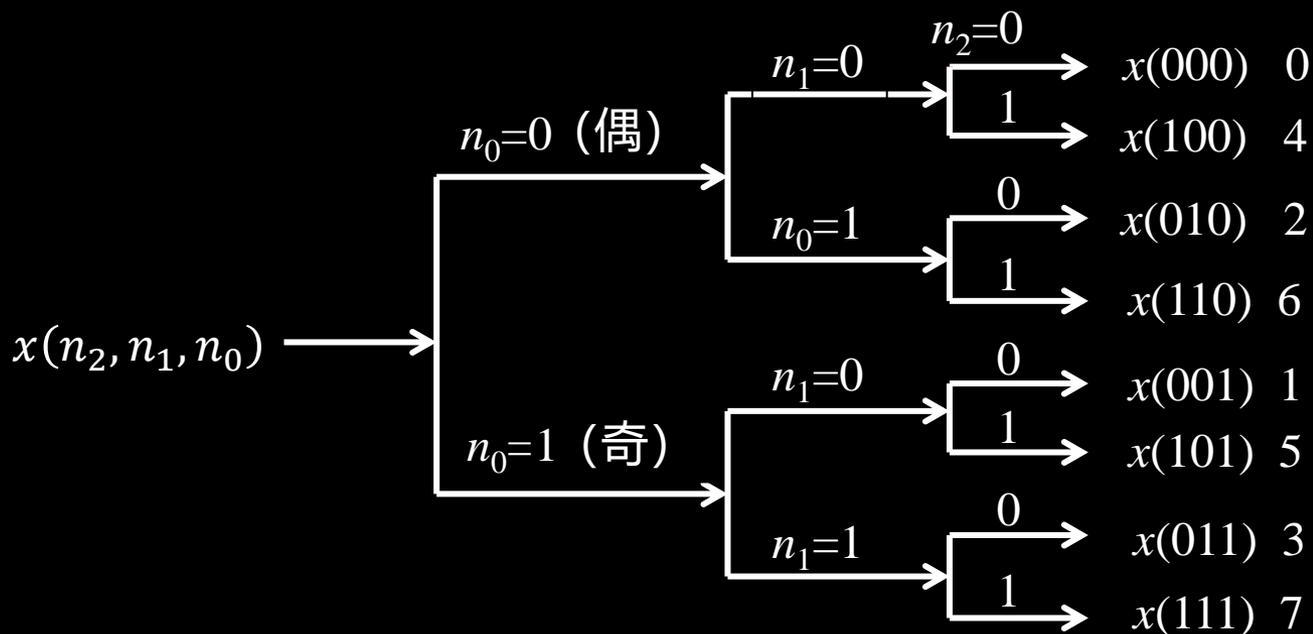


## ■ 倒位序的实现

由上图可知，按时域抽取FFT的输出 $X(k)$ 按自然顺序排列在存储单元，而输入是按以下顺序排列在存储单元：

$$x(0), x(4), x(2), x(6); x(1), x(5), x(3), x(7)$$

这种顺序称作**倒位序**，它是由奇偶分组造成的：



## ■ 码位倒置与自然序号的关系

对  $N=8=2^3$  的输入序列，序号二进制为  $(n_2, n_1, n_0)_2$ ，其倒位序二进制为  $(n_0, n_1, n_2)_2$

### 码位倒置与自然序号的关系

自然顺序	二进制 $n_2n_1n_0$	倒位序二进制 $n_0n_1n_2$	倒位顺序
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

## 5.3 按频率抽取的FFT算法

### ■ 算法原理 (基2FFT)

1、设  $N=2^M$ ，把输入序列  $x(n)$  按前后两部分分解成两个序列

$$\begin{aligned}
 X(k) &= \sum_{n=0}^{N-1} x(n)W_N^{nk} \\
 &= \sum_{n=0}^{N/2-1} x(n)W_N^{nk} + \sum_{n=N/2}^{N-1} x(n)W_N^{nk} \quad \text{前后分组} \\
 &= \sum_{n=0}^{N/2-1} x(n)W_N^{nk} + \sum_{n=0}^{N/2-1} x\left(n + \frac{N}{2}\right)W_N^{(n+N/2)k} \\
 &= \sum_{n=0}^{N/2-1} \left[ x(n) + x\left(n + \frac{N}{2}\right)W_N^{(N/2)k} \right] W_N^{nk}
 \end{aligned}$$

(每一个和式中出现  
的都是  $W_N^{nk}$ ，每一  
个和式都是  $N$  点  
DFT)

由于  $W_N^{\frac{N}{2}k} = \left(W_N^{\frac{N}{2}}\right)^k = (-1)^k$

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

## 2、分别计算 $k$ 为偶数和奇数的频率样本

当  $k$  为偶数, 即  $k=2r$  时, 有  $(-1)^k = 1$

当  $k$  为奇数, 即  $k=2r+1$  时, 有  $(-1)^k = -1$

$k$  为偶数时

$$\begin{aligned} X(2r) &= \sum_{n=0}^{N/2-1} \left[ x(n) + x\left(n + \frac{N}{2}\right) \right] W_N^{2nr} \\ &= \sum_{n=0}^{N/2-1} \left[ x(n) + x\left(n + \frac{N}{2}\right) \right] W_{N/2}^{nr} \end{aligned}$$

$k$  为奇数时

$$\begin{aligned} X(2r+1) &= \sum_{n=0}^{N/2-1} \left[ x(n) - x\left(n + \frac{N}{2}\right) \right] W_N^{n(2r+1)} \\ &= \sum_{n=0}^{N/2-1} \left\{ \left[ x(n) - x\left(n + \frac{N}{2}\right) \right] W_N^n \right\} W_{N/2}^{nr} \end{aligned}$$

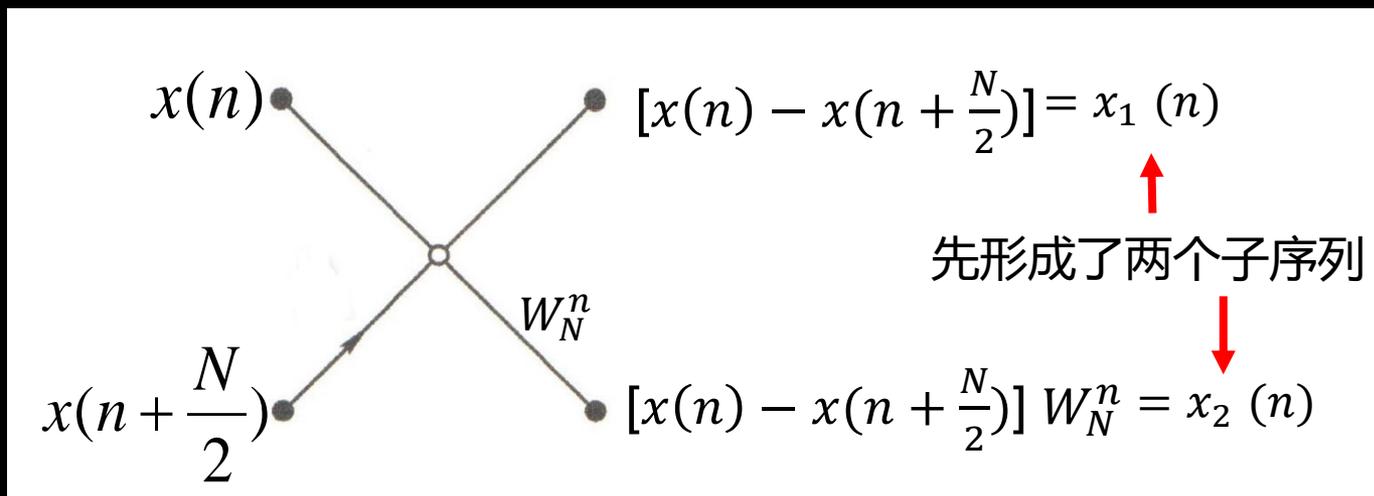
上面两式均满足  $N/2$  点 DFT 的定义式

(因为  $W_N^{2nr} = W_{N/2}^{nr}$ )

### 3、蝶形运算

$$\begin{cases} X(2r) = \sum_{n=0}^{\frac{N}{2}-1} \left[ x(n) + x\left(n + \frac{N}{2}\right) \right] W_{\frac{N}{2}}^{nr} & x_1(n) \\ X(2r+1) = \sum_{n=0}^{\frac{N}{2}-1} \left\{ \left[ x(n) - x\left(n + \frac{N}{2}\right) \right] W_N^n \right\} W_{\frac{N}{2}}^{nr} & x_2(n) \end{cases} \quad 0 \leq n \leq \frac{N}{2}-1$$

对 $x(n)$ 和 $x(n + \frac{N}{2})$ 进行如下蝶形运算



4、 $N=8$ 时的计算流程图（先形成子序列，得到两个4点的DFT运算）

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

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

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

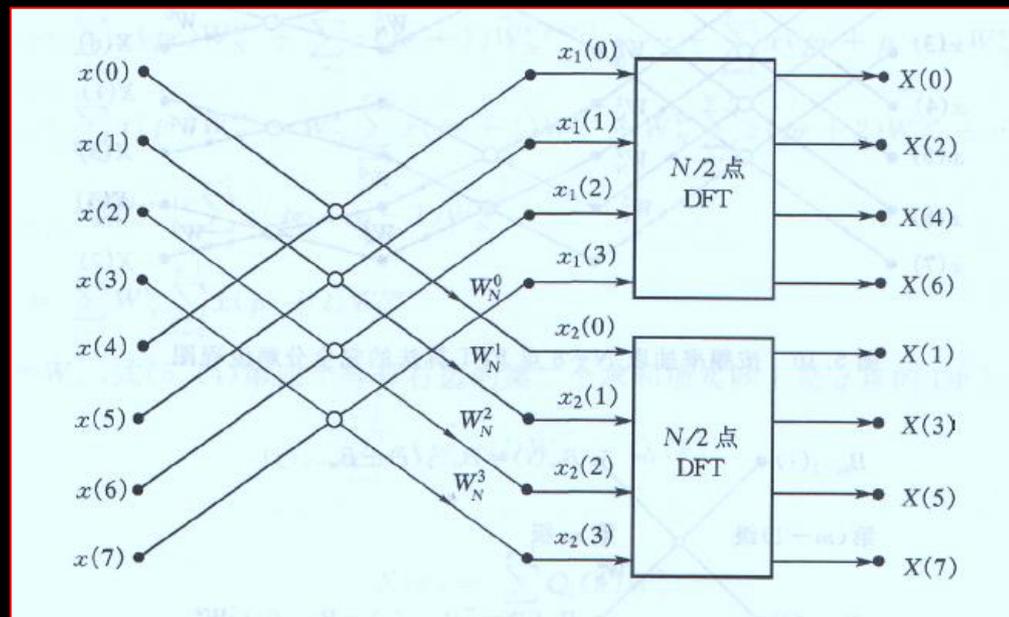
$$x_2(n) = \left[ x(n) - x\left(n + \frac{N}{2}\right) \right] W_N^n$$

子序列  $0 \leq n \leq \frac{N}{2} - 1$

因为  $W_N^2 = W_{\frac{N}{2}}$ ，于是，由上面两组公式得到两个  $\frac{N}{2}$  点的DFT运算，即

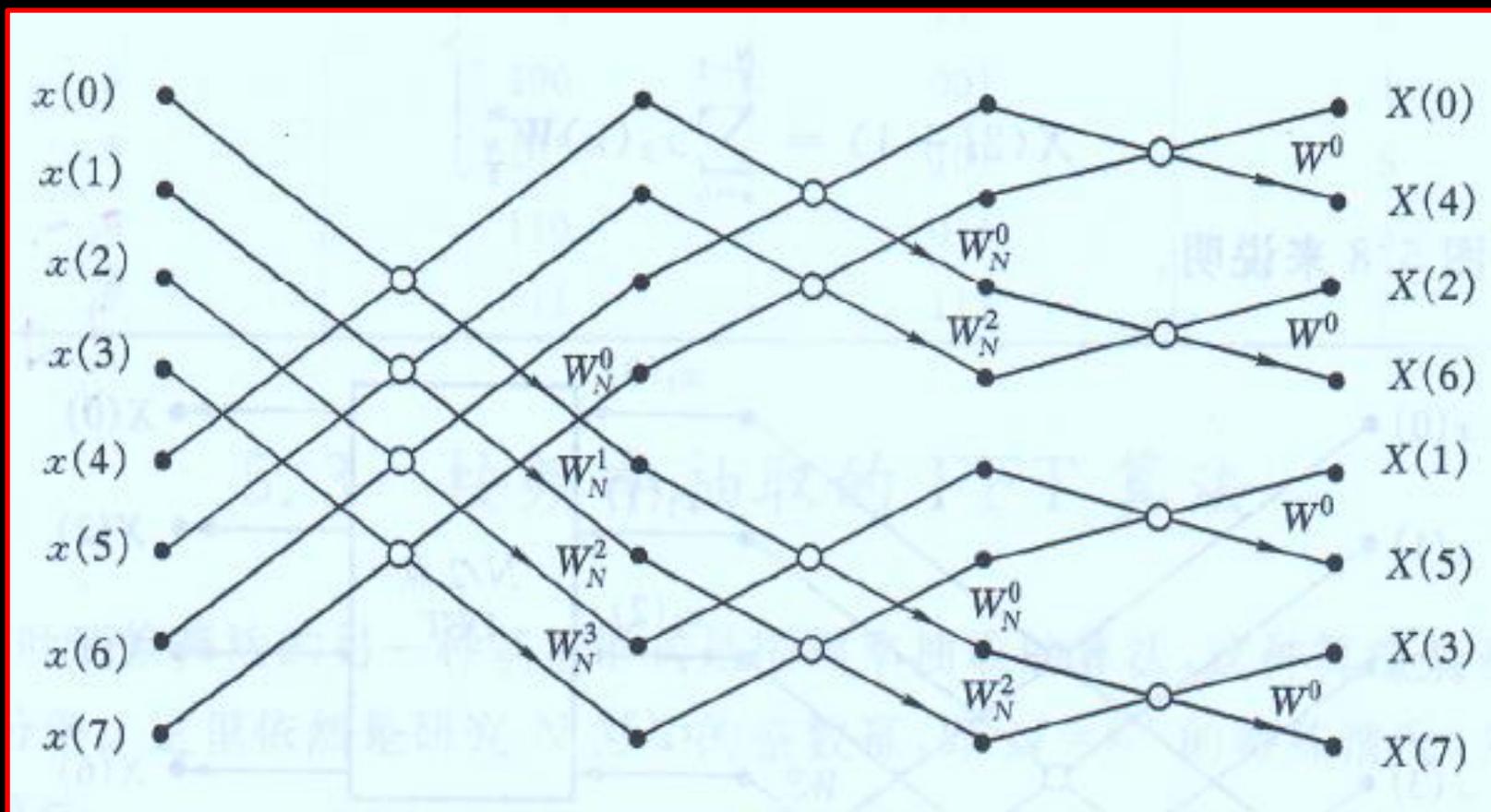
$$X(2r) = \sum_{n=0}^{\frac{N}{2}-1} x_1(n) W_{\frac{N}{2}}^{rn}$$

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



5、按照时域抽取FFT的思路，再将 $N/2$ 点DFT按 $k$ 的奇偶进一步分解为两个 $N/4$ 点的DFT，如此迭代进行下去，直至分解为单点DFT

得到  $N=8$  时按频域抽取FFT的完整计算流程图：



## ■ 权函数 $W_N^p$ 的确定

设迭代蝶形运算的次数为  $M$  ( $1 \leq l \leq M$ ) 次, 计算序列的长度为  $N=2^M$ ,  $l$  表示计算阵列的第  $l$  列

权函数 $W_N^p$ 的计算主要是确定  $p$  值,  $p$  值计算的一种方法如下:

- 1、把  $p$  值表示成  $M$  位的二进制数  $(p)_2$ , 其中  $M$  应满足:  $M=\log_2 N$ ,  $N$  为处理点数
- 2、将  $(p)_2$  右移  $(M-l)$  位, 并把左边的空位补零, 结果依然为  $M$  位
- 3、将移位补零的  $M$  位二进制数进行比特倒置
- 4、倒置后的二进制数转换成十进制数即得到  $p$  值

□ 一节点的权函数是 $W_N^p$ , 其对偶节点的权函数必然为 $W_N^{p+N/2}$ , 而且 $W_N^p = -W_N^{p+N/2}$ , 所以对偶节点可按下式计算

$$\begin{cases} x_l(k) = x_{l-1}(k) + W_N^p x_{l-1}(k + N/2^l) \\ x_l(k + N/2^l) = x_{l-1}(k) - W_N^{p+N/2} x_{l-1}(k + N/2^l) \end{cases}$$

## ■ 两种FFT的主要异同

### 1、不同点

**倒位序不同**：按时域抽取FFT的输入为倒位序，输出为自然顺序；按频域抽取FFT的输入为自然顺序，输出为倒位序

**蝶形运算形式不同**

### 2、相同点

**运算量相同**，均为 $(N/2)\text{Log}_2N$ 次复乘， $N\log_2N$ 次复加；

两种FFT形式上具有左右（输入、输出）对称性

## 5.4 IDFT的快速运算方法

### ■ 算法原理

$$\begin{aligned} X(k) &= \text{DFT}[x(n)] = \sum_{n=0}^{N-1} x(n)W_N^{nk} \\ x(n) &= \text{IDFT}[X(k)] = \frac{1}{N} \sum_{k=0}^{N-1} X(k)W_N^{-nk} \end{aligned}$$

比较两式可知，只要DFT的每个系数 $W_N^{nk}$ 换成 $W_N^{-nk}$ ，最后再乘以常数 $1/N$ 就可以得到IDFT的快速算法——IFFT

可以将常数 $1/N$ 分配到每一级蝶形运算中， $1/N=(1/2)^L$ ，即每级蝶形运算均乘以 $1/2$

## ■ 不改FFT程序直接实现 IFFT

由于  $[W_N^{-nk}]^* = W_N^{nk}$ ,  $[A \cdot B]^* = A^* \cdot B^*$

对IDFT取共轭, 得到

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

因此有

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

- 算法步骤:
- 1、先对 $X(k)$ 取共轭, 即将 $X(k)$ 的虚部乘 -1
  - 2、直接利用FFT程序计算其DFT
  - 3、对计算结果再取一次共轭
  - 4、最后再乘以常数 $1/N$ , 即得 $x(n)$

从而, FFT、IFFT 可共用同一个子程序

## 5.5 实数序列的FFT算法

通常考虑时间的实函数，而频率函数通常是复的，因此要设计一个即能减少计算DFT，又能计算IDFT的程序，就要假设一个复序列  $x(n)$  的FFT

$$X(k) = \sum_{n=0}^{N-1} [x_r(n) + jx_i(n)] e^{-j\frac{2\pi}{N}nk}$$

因为，根据复数共轭的反变换公式可以得到：

$$x(n) = \frac{1}{N} \left[ \sum_{k=0}^{N-1} [\operatorname{Re}(X(k)) + j\operatorname{Im}(X(k))]^* e^{-j\frac{2\pi}{N}nk} \right]^*$$

由于上面两式中，包含着共同的因子  $e^{-j\frac{2\pi}{N}nk}$ ，因此，用同一个程序就可以计算DFT和它的反变换

- 利用复时间函数的虚部，使实函数的FFT计算有更高的效率

## ■ 同时计算两个实序列的FFT

我们希望用下列复序列的形式：

$$v(n) = x(n) + jy(n)$$

同时计算两个实序列 $x(n)$ 和 $y(n)$ 的DFT。也就是说， $v(n)$ 是由两个实函数组成，其中一个实函数作为虚部

按DFT的线性性质， $v(n)$ 的DFT由下式给出

$$\begin{aligned} V(k) &= X(k) + jY(k) \\ &= [X_R(k) + jX_I(K)] + j[Y_R(k) + jY_I(K)] \\ &= \underline{[X_R(k) - Y_I(K)]} + j\underline{[X_I(k) + Y_R(K)]} \\ &= V_R(k) + jV_I(K) \end{aligned}$$

重写如下

$$V_R(k) = [X_R(k) - Y_I(k)], \quad V_I(k) = [X_I(k) + Y_R(k)]$$

由上式可以看到，求得的 $V(k)$ 的实部和虚部既含有的 $X(k)$ 成分，也有 $Y(k)$ 的成分，可以把 $V_R(k)$ 和 $V_I(k)$ 分解成奇偶序列，即：

$$V(k) = \left\{ \left[ \frac{V_R(k) + V_R(N-k)}{2} \right] + \left[ \frac{V_R(k) - V_R(N-k)}{2} \right] \right\} \\ + j \left\{ \left[ \frac{V_I(k) + V_I(N-k)}{2} \right] + \left[ \frac{V_I(k) - V_I(N-k)}{2} \right] \right\}$$

偶序列                      奇序列                      偶序列                      奇序列

### 根据DFT性质

若 $w(n)$ 是实序列，则  $W(k) = \underbrace{W_R(k)}_{\text{偶序列}} + j \underbrace{W_I(k)}_{\text{奇序列}}$

若 $w(n)$ 是纯虚序列，则  $W(k) = \underbrace{W_R(k)}_{\text{奇序列}} + j \underbrace{W_I(k)}_{\text{偶序列}}$

由上述的性质可以得出两个序列 $X(k)$ 和 $Y(k)$ 如下

$$X(k) = \left[ \frac{V_R(k) + V_R(N-k)}{2} \right] + j \left[ \frac{V_I(k) - V_I(N-k)}{2} \right] \quad (\text{A})$$

$$jY(k) = \left[ \frac{V_R(k) - V_R(N-k)}{2} \right] + j \left[ \frac{V_I(k) + V_I(N-k)}{2} \right]$$

或者

$$Y(k) = \left[ \frac{V_I(k) + V_I(N-k)}{2} \right] - j \left[ \frac{V_R(k) - V_R(N-k)}{2} \right] \quad (\text{B})$$

做一次 $N$ 点复序列的DFT，就能同时把两个 $N$ 点实序列的DFT求出来，因为求得 $V(k) = V_R(k) + jV_I(k)$ 后，按式(A)、(B)即可组合出 $X(k)$ 和 $Y(k)$ ，显然使运算效率提高一倍

## ■ 同时计算两个实序列的FFT的步骤

1. 函数 $x(n)$ 和 $y(n)$ 是实序列,  $n = 0, 1, \dots, N - 1$ 。
2. 构成复序列:  $v(n) = x(n) + jy(n)$ ,  $n = 0, 1, \dots, N - 1$
3. 计算

$$V(k) = \sum_{n=0}^{N-1} v(n)e^{-j2\pi nk/N} = V_R(k) + jV_I(k), \quad k = 0, 1, \dots, N - 1$$

式中 $V_R(k)$ 和 $jV_I(k)$ 分别为 $V(k)$ 的实部和虚部

4. 计算

$$\begin{cases} X(k) = \frac{1}{2}[V_R(k) + V_R(N-k)] + j\frac{1}{2}[V_I(k) - V_I(N-k)] \\ Y(k) = \frac{1}{2}[V_I(k) + V_I(N-k)] - j\frac{1}{2}[V_R(k) - V_R(N-k)] \end{cases} \quad k = 0, 1, \dots, N - 1$$

这里 $X(k)$ 和 $Y(k)$ 分别为 $x(n)$ 和 $y(n)$ 的DFT

## ■ 用 $N$ 点变换计算 $2N$ 点实序列的FFT

对长度为 $2N$ 点的实数序列 $x(n)$ , 其 $2N$ 点的DFT变换为

$$X(k) = \sum_{n=0}^{2N-1} x(n)W_{2N}^{kn}, \quad k = 0, 1, 2, \dots, 2N - 1$$

把 $x(n)$ 按 $n$ 的奇偶分开, 即

$$\begin{cases} x(2r) = u(r) \\ x(2r+1) = v(r) \end{cases} \quad r = 0, 1, 2, \dots, N - 1$$

其中 $U(k)$ 、 $V(k)$ 是两个长度为 $N$ 的实序列 $u(r)$ 、 $v(r)$ 的DFT, 可以通过前面讨论的“同时计算两个实序列的FFT的方法”实现

$$\begin{aligned} X(k) &= \sum_{r=0}^{N-1} x(2r)W_{2N}^{k(2r)} + \sum_{r=0}^{N-1} x(2r+1)W_{2N}^{k(2r+1)} = \sum_{r=0}^{N-1} u(r)W_{2N}^{kr} + W_{2N}^k \sum_{r=0}^{N-1} v(r)W_{2N}^{kr} \\ &= U(k) + W_{2N}^k V(k), \quad k = 0, 1, 2, \dots, 2N - 1 \end{aligned}$$

式中的 $k$ 是从 $0$ 到 $2N - 1$ , 但由于 $U(k+N) = U(k)$ 、 $V(k+N) = V(k)$ , 只需计算 $0 \leq k \leq N - 1$ 的 $U(k)$ 和 $V(k)$ , 又由于 $N \leq k \leq 2N - 1$ 时, 有 $W_{2N}^k (= e^{-j\pi k/N}) = -W_{2N}^{k+N} (= -e^{-j\pi(k+N)/N})$ , 所以当 $N \leq k \leq 2N - 1$ 时, 令 $k = N + l$ , 则

$$\begin{aligned} X(k) &= X(N + l) = U(l) + W_{2N}^{(l+N)} V(l) \\ &= U(l) - W_{2N}^l V(l), \quad l = 0, 1, 2, \dots, N - 1 \end{aligned}$$

## 5.6 利用FFT计算线性卷积和线性相关

### ■ 利用FFT计算线性卷积

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

其中,  $x(n)$ 长度为 $L$ ,  $h(n)$ 长度为 $M$ ,  $L \geq M$

#### 计算步骤:

- 1、分别对 $x(n)$ 、 $h(n)$ 补零点至长度至少为  $N=M+L-1$  点
- 2、用FFT求  $H(k)=\text{FFT}[h(n)]$
- 3、用FFT求  $X(k)=\text{FFT}[x(n)]$
- 4、求  $Y(k)=X(k)H(k)$
- 5、用IFFT求  $y(n)=\text{IFFT}[Y(k)]$

## ■ 利用FFT计算线性相关

$$\begin{aligned}
 r_{xy}(m) &= \sum_{n=0}^{L-1} x(n+m)y^*(n) = \sum_{n=0}^{L-1} x(n)y^*(n-m) \\
 &= x(m) * y^*(-m) \quad \text{(把线性相关转换到卷积表示)}
 \end{aligned}$$

其中,  $x(n)$ 长度为 $L$ ,  $y(n)$ 长度为 $M$ ,  $L \geq M$

### 计算步骤:

- 1、将 $x(n)$ 、 $y(n)$ 补零点至长度至少为  $N=M+L-1$  点
- 2、用FFT求  $X(k)=\text{FFT}[x(n)]$
- 3、用FFT求  $Y(k)=\text{FFT}[y(n)]$
- 4、求  $R_{xy}(k)=X(k)Y^*(k)$
- 5、用IFFT求  $r_{xy}(n)=\text{IFFT}[R_{xy}(k)]$

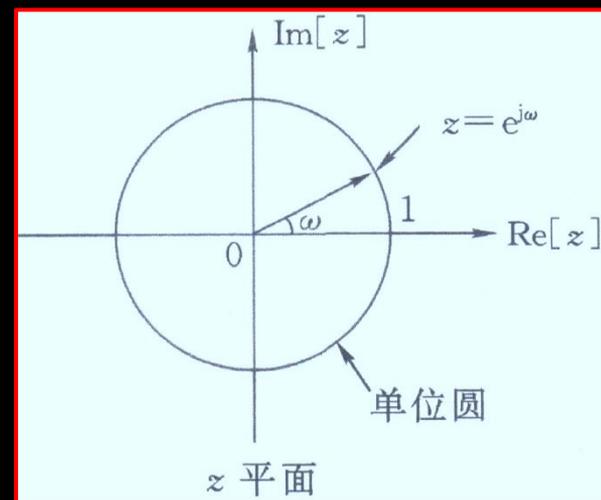
## 5.7 Chirp- $z$ 变换 (CZT)

### ■ 回顾：DFT与标准 $z$ 变换的关系

$z$ 变换  $Z[x(n)] = X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n}$

DFT  $X(k) = X(z)|_{z=e^{j\frac{2\pi}{N}k}=W_N^{-k}} = X(e^{j\frac{2\pi}{N}k}) = \sum_{n=0}^{N-1} x(n)W_N^{kn}$

- ✓ 在 $z$ 变换中,  $z = re^{j\omega}$ , 取单位圆  $r=1$ , 对单位圆进行等间隔采样  $\omega = \frac{2\pi}{N}k$ , 并取结果的主值区间  $k=[0, N-1]$ , 得  $z = e^{j\frac{2\pi}{N}k} = W_N^{-k}$ , 即得DFT
- ✓ 在实际中, 有时对一个时间序列的某个频率分段感兴趣, 可以使这个分段的采样频率与其它分段不一样



## ■ Chirp-z变换 (CZT) 的定义

z变换

$$Z[x(n)] = X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n}$$

在z变换中, 使z沿一段螺线作等角采样, 即采样点:

$$z_k = AW^{-k}, \quad k = 0, \dots, M-1$$

其中  $A = A_0 e^{j\theta_0}$ ,  $W = W_0 e^{-j\phi_0}$

参数含义:

$A_0$ ——起始采样点的矢量半径

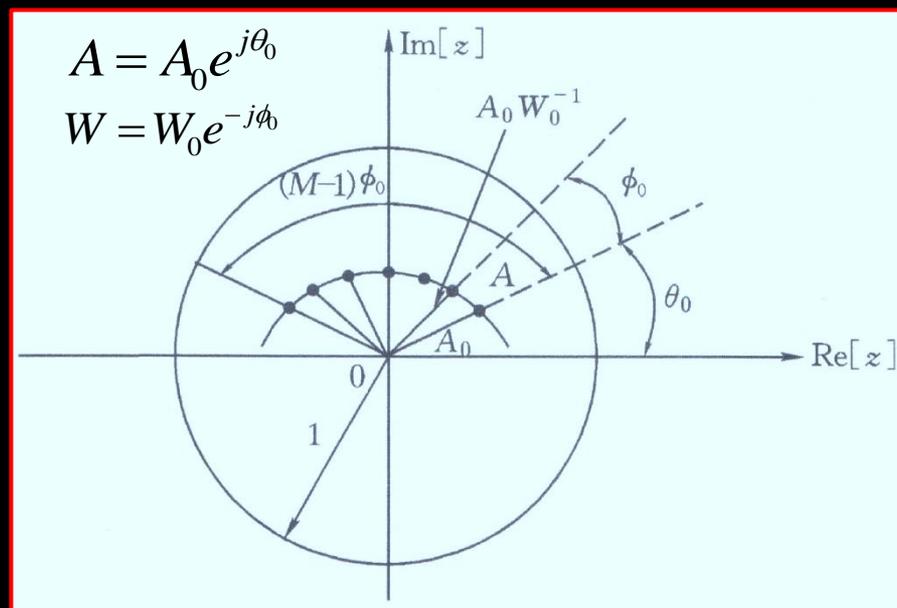
$\theta_0$ ——起始采样点的相角

$\phi_0$ ——两相邻采样点间的角度差

$W_0$ ——螺线的伸展率

$W_0 > 1$ 时, 螺线内缩

$W_0 < 1$ 时, 螺线外伸



对Chirp- $z$ 变换  $z_k = AW^{-k}$ ,  $k = 0, \dots, M - 1$ , 可以看出当取  $A = 1$ ,  $W = e^{-j\frac{2\pi}{N}}$  时, 有  $z_k = e^{j\frac{2\pi}{N}k}$ ,  $k = 0, \dots, M - 1$ , Chirp- $z$ 变换即退化成为 DFT

### ■ CZT的优势

- 1、可以只计算单位圆上感兴趣的一小段频谱的采样, 而非整个单位圆, 这特别适用于高分辨率窄带信号
- 2、可以计算远离单位圆的任意点处的频谱, 特别适用于语音及雷达信号

## ■ CZT的快速算法

### CZT的表达式

$$X(z_k) = \sum_{n=0}^{N-1} x(n)z_k^{-n} = \sum_{n=0}^{N-1} x(n)A^{-n}W^{nk},$$

$$k = 0, \dots, M-1$$

得到↓

$$X(z_k) = W^{\frac{k^2}{2}} \sum_{n=0}^{N-1} \left[ x(n)A^{-n}W^{\frac{n^2}{2}} \right] W^{-\frac{(k-n)^2}{2}}$$

得到↓

$$X(z_k) = W^{\frac{k^2}{2}} \sum_{n=0}^{N-1} g(n)h(k-n) = W^{\frac{k^2}{2}} [g(k) * h(k)]$$

用FFT求解线性卷积即可实现CZT的快速计算

### Bluestein等式

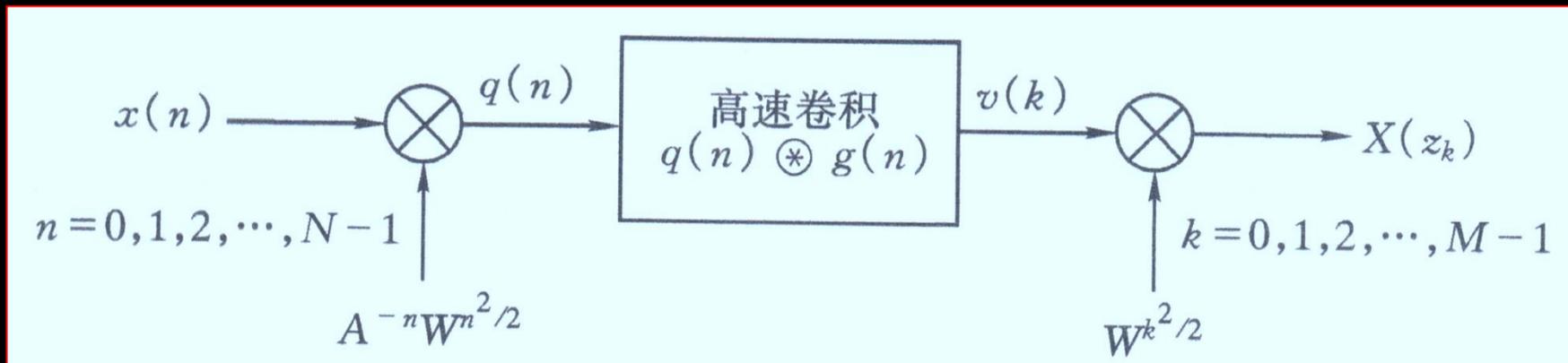
$$kn = \frac{1}{2} \left[ n^2 + k^2 - (k-n)^2 \right]$$

令

$$q(n) = x(n)A^{-n}W^{\frac{n^2}{2}}$$

$$g(n) = W^{-\frac{n^2}{2}}$$

## ■ 基于卷积计算的CZT变换的运算过程



卷积的长度  $L \geq N + M - 1$ , 若取  $L = N + M - 1$ , 则有

$$q(n) = \begin{cases} x(n)A^{-n}W^{n^2/2}, & n = 0, 1, 2, \dots, N-1 \\ 0, & n = N, N+1, \dots, L-1 \end{cases}$$

$$g(n) = \begin{cases} W^{-n^2/2}, & 0 \leq n \leq M-1 \\ W^{-(L-n)^2/2}, & L-N+1 \leq n < L \\ \text{任意值} & \text{其他 } n \end{cases}$$

## 本章小结

- FFT算法的基本原理
- 基2 FFT算法的理论推导
- 按时间抽取的FFT算法
- 按频率抽取的FFT算法
- IDFT的快速运算方法
- FFT在线性卷积、相关中的应用