

第 4 章 快速傅里叶变换

笪邦友

中南民族大学 电子信息工程学院

2018 年 10 月 30 日

目录

1 引言

2 基 2FFT 算法

- 直接计算 DFT 的特点及减少运算量的基本途径
- 时域抽取法基 2FFT 基本原理 (DIT-FFT)
- DIT-FFT 算法计算量

3 DIT-FFT 的运算规律与编程思想

- 原位计算 (同址计算)
- 旋转因子 W_N^p 的变化规律
- 蝶形运算规律
- 编程思想及程序框图
- 序列的倒序
 - 倒序的概念
 - 倒序的编程思想及程序框图

4 频域抽取法 (DIF-FFT)

- 频域抽取法 (DIF-FFT)

5 IDFT 的快速算法

引言

- DFT 是一种针对有限长带限信号的傅里叶变换理论。但实践中不存在这样的信号，对于时域有限的信号，其频域为无限长，在不影响信号分析的情况下，可通过滤除高于折叠频率的频率成分，使之频域成为有限长。这一点称为 DFT 理论上的近似性。
- 1965 年，库利和图基在《计算数学》上发表著名论文《机器计算傅里叶级数的一种算法》后很快形成了一套高效计算方法，就是现在的快速傅里叶变换算法。FFT 是实现 DFT 的一种快速算法，其在理论上没有任何误差。
- FFT 算法的提出，标志着数字信号处理这门学科的开端。

直接计算 DFT 的特点

有限长序列 $x(n)$ 的 N 点 DFT 为 $X(k)$, 则有:

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

分析方程可知:

输入信号为:

$x(0), x(1), x(2), \dots, x(N-1)$

那么频域变换结果为:

$X(0), X(1), X(2), \dots, X(N-1)$

N 点 DFT 的计算量

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

令 $k = 2$, 则

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

- ① 显然, 计算 $X(k)$ 的 1 个值, 需要做 $N-1$ 次加法, N 次乘法
- ② 那么, 计算 $X(k)$ 的所有 N 个值, 则需要 N^2 次复数乘法,
 $N(N-1) \approx N^2$ 次复数加法。
- ③ 如 $N = 2^{10}$, 则 $N^2 = 1048576$

可见直接计算 DFT 计算量很大。

减少计算 DFT 运算量的途径

① 基本思想：将一个长序列 DFT 分解为若干个短序列 DFT。

- 如 $N = 2^{10} = 1024$, 可分为两个长度为 512 的短序列。
- 一个短序列的乘法次数为 $(\frac{N}{2})^2$,
- 则两个短序列的乘法计算时间为, $(\frac{N}{2})^2 + (\frac{N}{2})^2 = \frac{N^2}{2}$

② 问题在于：

- ① 将长序列分离为短序列可行吗？
- ② 两个短序列的 DFT 合并后与长序列的 DFT 等价吗？为什么？

旋转因子 ($W_N = e^{-j\frac{2\pi}{N}}$) 的性质

短序列 DFT 与长序列 DFT 的关系的关键在于旋转因子的性质

① 周期性: $W_N^{m+lN} = W_N^m$

$$(W_N^{m+lN} = e^{-j\frac{2\pi}{N}(m+lN)} = e^{-j\frac{2\pi}{N}m} \cdot e^{-j2\pi l} = W_N^m)$$

② 对称性 1: $W_N^{m+\frac{N}{2}} = -W_N^m$ ($W_N^{\frac{N}{2}} = e^{-j\frac{2\pi}{N}\frac{N}{2}} = e^{-j\pi} = -1$)

③ 对称性 2: $W_N^{2kn} = W_{N/2}^{kn}$

$$(W_N^{2kn} = e^{-j\frac{2\pi}{N}2kn} = e^{-j\frac{2\pi}{N/2}kn} = W_{N/2}^{kn})$$

时域抽取法基 2FFT 基本原理 (DIT-FFT)

时域抽取法基 2FFT 将长序列分解为短序列所遵循的原则

- ① 对时间奇偶分 (n)
- ② 对频率前后分 (k)

(一) 第一次分解

假设序列 $x(n)$ 长度为 N , 且满足 $N = 2^M$, M 为自然数。

按 n 的奇偶把 $x(n)$ 分解为两个长度为 $N/2$ 的子序列

$$\begin{cases} x_1(r) = x(2r) & r = 0, 1, \dots, \frac{N}{2} - 1 \\ x_2(r) = x(2r + 1) & r = 0, 1, \dots, \frac{N}{2} - 1 \end{cases}$$

注意: 此处存在 3 个序列,

- 原序列 $x(n)$ 长度为 N
- 两个子序列 $x_1(r)$ 、 $x_2(r)$, 长度分别为 $\frac{N}{2}$

以 $N=8$ 的数字序列为例

例如: $N = 2^3 = 8$ 则:

$$x(n) = \{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}$$

可分解为两个长度为 $\frac{N}{2}$ 的短序列,

$$\begin{cases} x_1(r) = x(2r) & = \{x(0), x(2), x(4), x(6)\} \text{ 序列 } x(n) \text{ 的偶数部分} \\ x_2(r) = x(2r+1) & = \{x(1), x(3), x(5), x(7)\} \text{ 序列 } x(n) \text{ 的奇数部分} \end{cases}$$

注意, 这里 $0 \leq r \leq \frac{N}{2} - 1$

推导：将 $X(k)$ 分解为两个短序列的 DFT

$$\begin{aligned}
 X(k) &= DFT[x(n)] \quad (0 \leq k \leq N-1) \\
 &= \sum_{n=0}^{N-1} x(n) W_N^{kn} \\
 &= \sum_{\substack{n \text{ 为偶数}}} x(n) W_N^{kn} + \sum_{\substack{n \text{ 为奇数}}} x(n) W_N^{kn} \\
 &= \sum_{r=0}^{\frac{N}{2}-1} x(2r) W_N^{k2r} + \sum_{r=0}^{\frac{N}{2}-1} x(2r+1) W_N^{k(2r+1)} \\
 &= \sum_{r=0}^{\frac{N}{2}-1} x(2r) W_N^{2kr} + \sum_{r=0}^{\frac{N}{2}-1} x(2r+1) W_N^{2kr} W_N^k \\
 &= \sum_{r=0}^{\frac{N}{2}-1} x_1(r) W_{\frac{N}{2}}^{kr} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x_2(r) W_{\frac{N}{2}}^{kr} \quad (\text{因为: } W_N^{2kr} = W_{N/2}^{kr})
 \end{aligned}$$

长序列傅里叶变换和短序列傅里叶变换的关系?

$$\text{即: } X(k) = \sum_{r=0}^{\frac{N}{2}-1} x_1(r) W_N^{kr} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x_2(r) W_N^{kr}$$

设:

$$\begin{cases} X_1(k) = DFT[x_1(r)] = \sum_{r=0}^{\frac{N}{2}-1} x_1(r) W_N^{kr} & (0 \leq r \leq \frac{N}{2} - 1) \\ X_2(k) = DFT[x_2(r)] = \sum_{r=0}^{\frac{N}{2}-1} x_2(r) W_N^{kr} & (0 \leq r \leq \frac{N}{2} - 1) \end{cases}$$

那么是否有:

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

长序列傅里叶变换和短序列傅里叶变换的关系？

对于等式

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

存在一个问题：

$X(k)$ 长度为 N ，而 $X_1(k)$ 与 $X_2(k)$ 长度均为 $N/2$ 。

所以不能直接给出上述结论，需分两种情况讨论。

(1) 当 $0 \leq k \leq \frac{N}{2} - 1$ 时，

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

(2) 当 $\frac{N}{2} \leq k \leq N - 1$ 时，不能直接给出上式，因为上式右边的短序列长度为 $\frac{N}{2}$ ，可令 $k = k_1 + \frac{N}{2}$ ，则 $0 \leq k_1 \leq \frac{N}{2} - 1$ ，然后在对原式进行推导。

分两种情况讨论

(1) 当 $0 \leq k \leq \frac{N}{2} - 1$ 时, $X(k) = X_1(k) + W_N^k \cdot X_2(k)$

(2) 当 $\frac{N}{2} \leq k \leq N - 1$ 时, 令 $k = k_1 + \frac{N}{2}$, 则 $0 \leq k_1 \leq \frac{N}{2} - 1$ 。

$$\begin{aligned} X\left(k_1 + \frac{N}{2}\right) &= \sum_{r=0}^{\frac{N}{2}-1} x_1(r) W_{\frac{N}{2}}^{(k_1 + \frac{N}{2})r} + W_N^{(k_1 + \frac{N}{2})} \sum_{r=0}^{\frac{N}{2}-1} x_2(r) W_{\frac{N}{2}}^{(k_1 + \frac{N}{2})r} \\ &= \sum_{r=0}^{\frac{N}{2}-1} x_1(r) W_{\frac{N}{2}}^{k_1 r} - W_N^{k_1} \sum_{r=0}^{\frac{N}{2}-1} x_2(r) W_{\frac{N}{2}}^{k_1 r} \quad (0 \leq k_1 \leq \frac{N}{2} - 1) \end{aligned}$$

将 k_1 替换为 k , 则有:

$$\begin{aligned} X\left(k + \frac{N}{2}\right) &= \sum_{r=0}^{\frac{N}{2}-1} x_1(r) W_{\frac{N}{2}}^{kr} - W_N^k \sum_{r=0}^{\frac{N}{2}-1} x_2(r) W_{\frac{N}{2}}^{kr} \quad (0 \leq k \leq \frac{N}{2} - 1) \\ &= X_1(k) - W_N^k X_2(k) \end{aligned}$$

蝶形运算

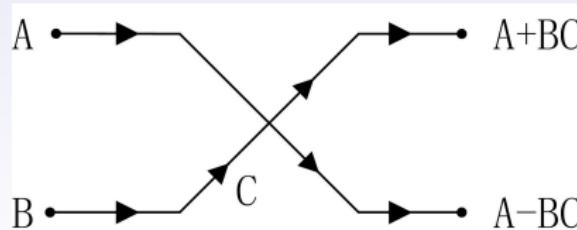
$$\begin{cases} X(k) = X_1(k) + W_N^k X_2(k) & (0 \leq k \leq \frac{N}{2} - 1) \\ X(k + \frac{N}{2}) = X_1(k) - W_N^k X_2(k) \end{cases}$$

上式成功的将一个 N 点的 DFT 转化为两个 N/2 点 DFT

蝶形运算

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

上式成功的将一个 N 点的 DFT 转化为两个 $N/2$ 点 DFT
这种计算方式很有规律，称为蝶形运算。



显然，一个蝶形运算，需要 1 次复数乘法，2 次复数加法。

以 8 点 DFT 为例

令 $k = 0$, 则有

$$\begin{cases} X(0) = X_1(0) + W_N^0 X_2(0) \\ X(4) = X_1(0) - W_N^0 X_2(0) \end{cases}$$

令 $k = 1$, 则有

$$\begin{cases} X(1) = X_1(1) + W_N^1 X_2(1) \\ X(5) = X_1(1) - W_N^1 X_2(1) \end{cases}$$

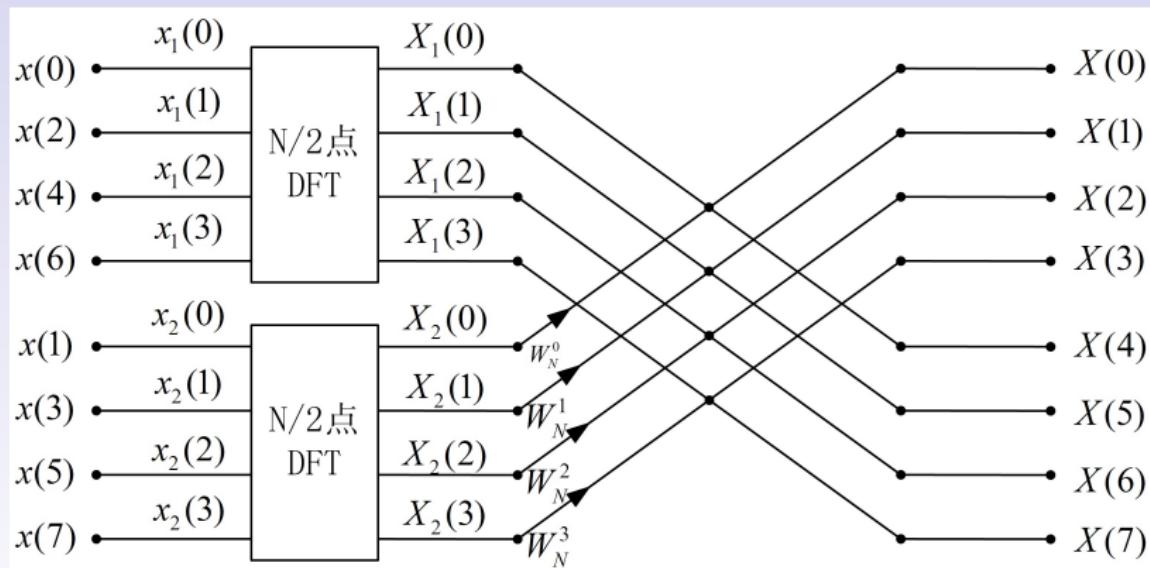
令 $k = 2$, 则有

$$\begin{cases} X(2) = X_1(2) + W_N^2 X_2(2) \\ X(6) = X_1(2) - W_N^2 X_2(2) \end{cases}$$

令 $k = 3$, 则有

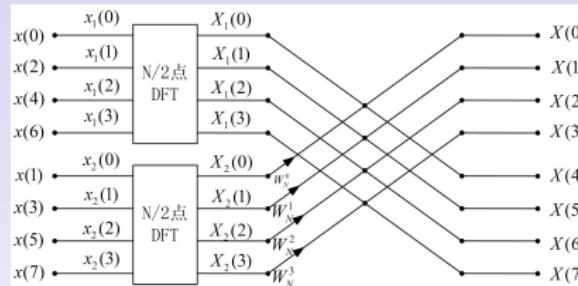
$$\begin{cases} X(3) = X_1(3) + W_N^3 X_2(3) \\ X(7) = X_1(3) - W_N^3 X_2(3) \end{cases}$$

8 点 DFT 一次时域抽取分解运算流图



一个 N 点 DFT \iff 两个 $\frac{N}{2}$ 点 DFT + $\frac{N}{2}$ 次蝶形运算

第一次分解后 FFT 计算量分析



一个 N 点 DFT \iff 两个 $\frac{N}{2}$ 点 DFT + $\frac{N}{2}$ 次蝶形运算

计算量分析：

- 两个短序列： $(\frac{N}{2})^2 \times 2 = \frac{N^2}{2}$
- $N/2$ 个蝶形： $N/2$ 个乘法
- 总乘法次数， $\frac{N^2}{2} + \frac{N}{2} \approx \frac{N^2}{2}$

可见，一次分解后，计算量减半。

(二) 第二次分解

经过第一次分解，将

$x(n) = \{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}$ 分解为两个长度为 $\frac{N}{2}$ 的短序列。

$$\begin{cases} x_1(r) = \{x(0), x(2), x(4), x(6)\} = x(2r) \\ x_2(r) = \{x(1), x(3), x(5), x(7)\} = x(2r+1) \end{cases}$$

设 $x_1(r) \longleftrightarrow X_1(k)$ 长 $\frac{N}{2}$

再次对 $x_1(r)$ 按奇偶分为两个 $N/4$ 点的子序列 $x_3(l)$ 、 $x_4(l)$ ，即

$$\begin{cases} x_3(l) = x_1(2l) & (0 \leq l \leq \frac{N}{4} - 1) \\ x_4(l) = x_1(2l + 1) \end{cases}$$

第二次分解——推导

显然, 对于 8 点 DFT, 此处有

$$\begin{cases} x_3(l) = \{x(0), x(4)\} \\ x_4(l) = \{x(2), x(6)\} \end{cases}$$

如果

$$\begin{cases} x_3(l) \longleftrightarrow X_3(k) & (0 \leq l \leq \frac{N}{4} - 1) \\ x_4(l) \longleftrightarrow X_4(k) \end{cases}$$

可将 $X_1(k)$ 用 $X_3(k), X_4(k)$ 表示。

与前述类似, 可类推下列公式。

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

第二次分解——示例

例如：

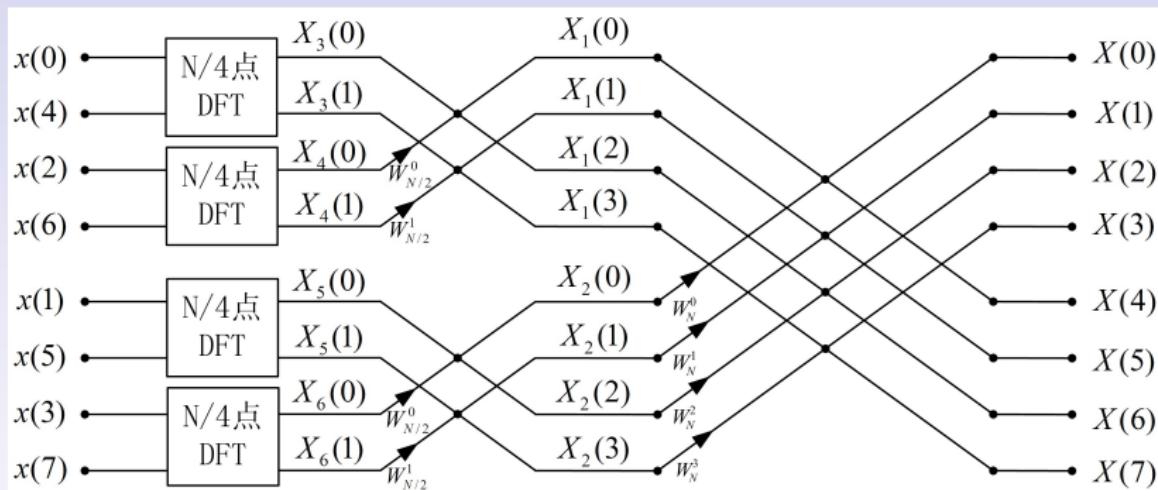
令 $k = 0$, 则有

$$\begin{cases} X_1(0) = X_3(0) + W_{\frac{N}{2}}^0 X_4(0) \\ X_1(2) = X_3(0) - W_{\frac{N}{2}}^0 X_4(0) \end{cases}$$

令 $k = 1$, 则有

$$\begin{cases} X_1(1) = X_3(1) + W_{\frac{N}{2}}^1 X_4(1) \\ X_1(3) = X_3(1) - W_{\frac{N}{2}}^1 X_4(1) \end{cases}$$

8 点 DFT 二次时域抽取分解运算流图



最后一次分解

一直分解到短序列长度为 2，不能再分。此时，DFT 计算按定义来算。

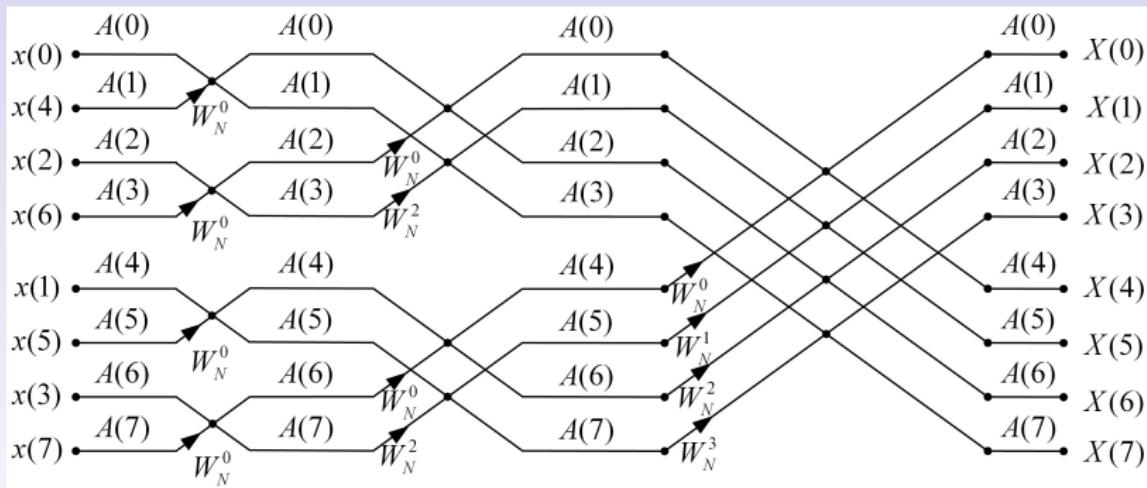
例如：当 $N=2$ 时，

$$\begin{aligned} X(k) &= \sum_{n=0}^1 x(n) W_2^{kn} \quad (0 \leq k \leq 1) \\ &= x(0) W_2^{k \cdot 0} + x(1) W_2^{k \cdot 1} \\ &= x(0) + x(1) W_2^k \quad (0 \leq k \leq 1) \end{aligned}$$

将 $k = 0, k = 1$ 代入可得：

$$\begin{cases} X(0) = x(0) + x(1) \\ X(1) = x(0) - x(1) \quad (W_2^1 = -1) \end{cases}$$

8 点 DFT 第三次时域分解运算流图



请注意：FFT 算法的运算流图全部是蝶形运算。

DIT-FFT 算法计算量分析

设 $N = 2^M$ ($M = \log_2 N$),

1 运算流图应有 M 级蝶形。

2 每一级都由 $\frac{N}{2}$ 个蝶形构成,

3 每个蝶形需要一次复数乘法, 两次复数加法,

因此, M 级蝶形运算总的复数乘法次数为: $M \cdot \frac{N}{2} = \frac{N}{2} \log_2 N$
以乘法为例:

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

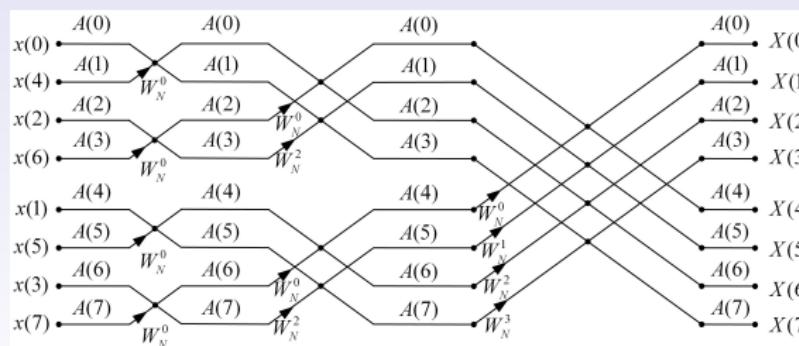
- 当 $N = 2^{10} = 1024$ 时, $\frac{DFT}{FFT} = 204.8$
- 当 $N = 2^{20} = 1024 \times 1024$ 时, $\frac{DFT}{FFT} = 104857.6$

可见, N 越大, 节约的时间越多。

原位计算

DIT-FFT 的运算过程很有规律

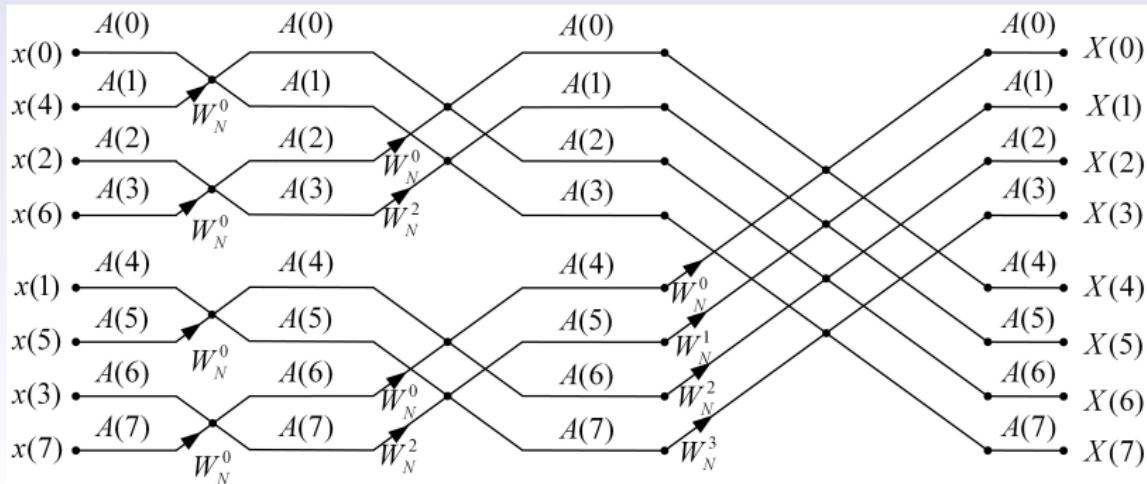
- 1 N 点的 FFT 共进行 M 级运算，每级有 $\frac{N}{2}$ 个蝶形；
- 2 同一级中，每个蝶形的两个输入数据只对计算本蝶形有用；
- 3 且输入输出处于同一条水平线上，这意味着计算完一个蝶形后，所输出数据可立即存入原输入数据所占用的存储单元。



这种利用同一存储单元存储蝶形计算输入输出的方法称为原位计算。可以节省大量内存，使设备成本降低。

旋转因子

- ① N 点 $DIT-DFT$ 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 。
- ② 这里称 W_N^p 为旋转因子， p 称为旋转因子的指数。



└ DIT-FFT 的运算规律与编程思想

 └ 旋转因子 W_N^p 的变化规律

旋转因子 W_N^p 与运算级数的关系

设 L 为从左到右的运算级数，即 $L = 1, 2, 3, \dots, M$, 有以下规律：

旋转因子 W_N^p 与运算级数的关系

设 L 为从左到右的运算级数，即 $L = 1, 2, 3, \dots, M$ ，有以下规律：

- 1 第 L 级一共有 2^{L-1} 个不同的旋转因子。
- 2 第 L 级中，

$$W_N^p = W_{2^L}^J, \quad J = 0, 1, \dots, (2^{L-1} - 1)$$

└ DIT-FFT 的运算规律与编程思想

└ 旋转因子 W_N^p 的变化规律

旋转因子 W_N^p 与运算级数的关系

设 L 为从左到右的运算级数, 即 $L = 1, 2, 3, \dots, M$, 有以下规律:

- 1 第 L 级一共有 2^{L-1} 个不同的旋转因子。
- 2 第 L 级中,

$$W_N^p = W_{2^L}^J, \quad J = 0, 1, \dots, (2^{L-1} - 1)$$

如: $L = 1$ 时, $W_N^p = W_2^J$ $J = 0$

推导:

$$W_{2^L}^J = W_{2^M \cdot 2^{L-M}}^J = W_{N \cdot 2^{L-M}}^J = W_N^{J/2^{L-M}} = W_N^{J \cdot 2^{M-L}} = W_N^p$$

└ DIT-FFT 的运算规律与编程思想

└ 旋转因子 W_N^p 的变化规律

旋转因子 W_N^p 与运算级数的关系

设 L 为从左到右的运算级数, 即 $L = 1, 2, 3, \dots, M$, 有以下规律:

- 1 第 L 级一共有 2^{L-1} 个不同的旋转因子。
- 2 第 L 级中,

$$W_N^p = W_{2^L}^J, \quad J = 0, 1, \dots, (2^{L-1} - 1)$$

如: $L = 1$ 时, $W_N^p = W_2^J$ $J = 0$

推导:

$$W_{2^L}^J = W_{2^M \cdot 2^{L-M}}^J = W_{N \cdot 2^{L-M}}^J = W_N^{J/2^{L-M}} = W_N^{J \cdot 2^{M-L}} = W_N^p$$

显然有:

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

└ DIT-FFT 的运算规律与编程思想

└ 旋转因子 W_N^p 的变化规律

以 $N=8$ 为例, 旋转因子为:

$$p = J \cdot 2^{M-L} \quad J = (0, 1, \dots, (2^{L-1} - 1))$$

例如: 8 点 DFT 时, $M=3$, 有

① $L = 1, \quad J = (0)$

- $p = J \cdot 2^{M-L} = J \cdot 4 = (0)$
- 旋转因子为: W_N^0

② $L = 2, \quad J = (0, 1)$

- $p = J \cdot 2^{M-L} = J \cdot 2 = (0, 2)$
- 旋转因子为: $W_N^0 \quad W_N^2$

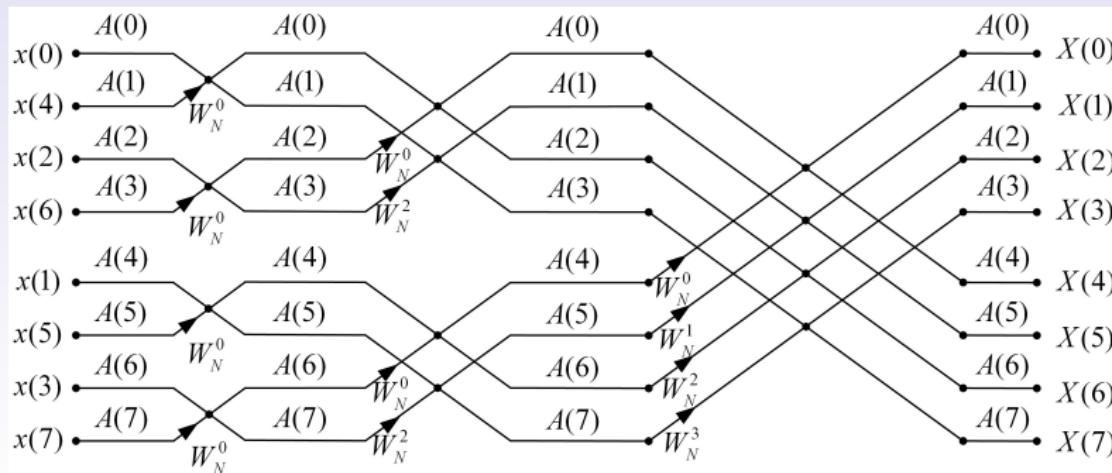
③ $L = 3, \quad J = (0, 1, 2, 3)$

- $p = J \cdot 2^{M-L} = J = (0, 1, 2, 3)$
- 旋转因子为: $W_N^0 \quad W_N^1 \quad W_N^2 \quad W_N^3$

蝶形运算规律

在第 L 级中，有

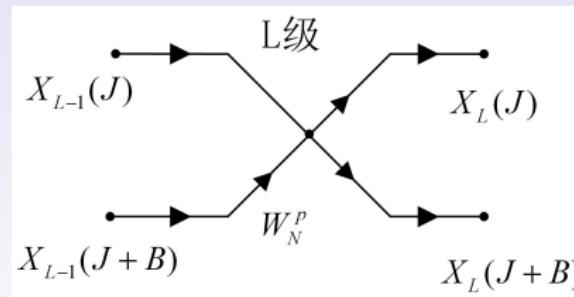
- 1 每个蝶形两个输入端相距点数为： $B = 2^{L-1}$
- 2 有 $B = 2^{L-1}$ 个不同的旋转因子
- 3 同一个旋转因子，对应相邻间隔为 2^L 点的 2^{M-L} 个蝶形



蝶形运算的计算公式

1 在第 L 级中，每个蝶形两个输入端相距点数为： $B = 2^{L-1}$

$$\begin{cases} A_{L-1}(J) + W_N^p A_{L-1}(J+B) \rightarrow A_L(J) \\ A_{L-1}(J) - W_N^p A_{L-1}(J+B) \rightarrow A_L(J+B) \end{cases}$$



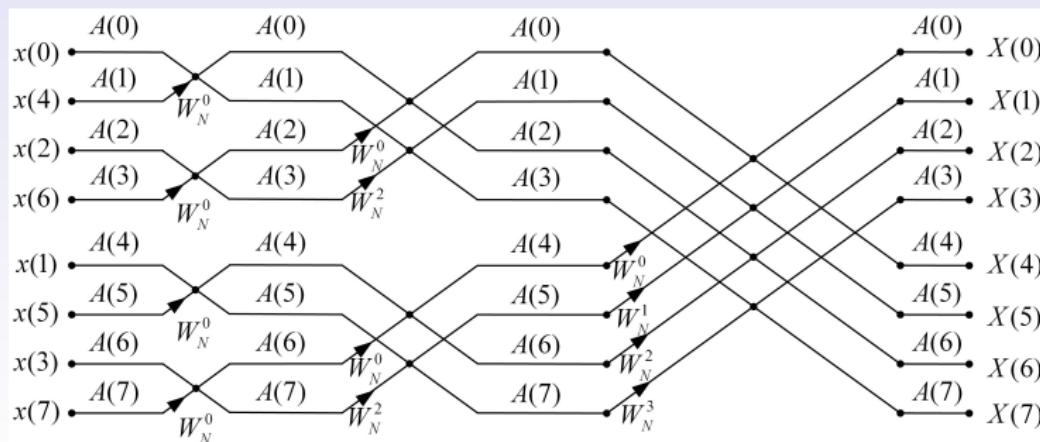
说明：容易混淆的情况

1 两个 2^{L-1}

(a) 第 L 级中，有 $B = 2^{L-1}$ 个不同的旋转因子

(b) 第 L 级中，每个蝶形的两个输入端相距的点数 $B = 2^{L-1}$

2 2^L : 同一级中具有同一旋转因子的不同蝶形相邻的点数.



总结：第 L 层旋转因子出现的规律

$N = 2^M$, 共有 M 级运算, 每级有 $N/2$ 个蝶形。

- 1 第 L 级中, 每个蝶形两个输入端相距点数为: $B = 2^{L-1}$
- 2 第 L 级中, 一共有 2^{L-1} 个不同的旋转因子
- 3 第 L 级中, 同一个旋转因子, 对应 2^{M-L} 个蝶形, 每个蝶形相邻间隔为 2^L 点

编程思想

那么将分三个循环进行：

- ① 进行 $L = 1, 2, \dots, M$ 级运算。(第一级循环)

$$L = 1, \dots, M$$

- ② 在第 L 级中，存在 $B = 2^{L-1}$ 个旋转因子，按旋转因子不同进行计算。(第二级循环)

$$B = 2^{L-1}$$

$$J = 0, \dots, B - 1$$

- ③ 每一个旋转因子对应的 2^{M-L} 个蝶形运算。(第三级循环)

$$k = J, N - 1, 2^L$$

(k 从 J 开始，每循环一次增加 2^L ，到不超过 $N-1$ 为止)

举例说明——以第二级为例, $L = 2$

- ① 计算该层的旋转因子个数, 以及每个蝶形相距的点数。

$$L = 2 \quad \therefore B = 2^{L-1} = 2$$

- ② 计算旋转因子的指数 p

$$J = (0, 1) \quad p = 2^{M-L} \cdot J = 2 \cdot J = (0, 2)$$

- ③ 每个旋转因子的蝶形计算, 同一旋转因子间隔为 $2^L = 4$

$$k = J, 7, 4$$

- ④ 计算公式

$$\begin{cases} A_{L-1}(k) + W_N^p A_{L-1}(k+B) \longrightarrow A_L(k) \\ A_{L-1}(k) - W_N^p A_{L-1}(k+B) \longrightarrow A_L(k+B) \end{cases}$$

示例——计算过程 I

$$L = 2 \quad \therefore B = 2^{L-1} = 2$$

1 $J = 0$

(a) $J = 0, p = 0, k = 0, B = 2$

$$\begin{cases} A(0) + W_N^0 A(2) \longrightarrow A(0) \\ A(0) - W_N^0 A(2) \longrightarrow A(2) \end{cases}$$

(b) $J = 0, p = 0, k = 4, B = 2$

$$\begin{cases} A(4) + W_N^0 A(6) \longrightarrow A(4) \\ A(4) - W_N^0 A(6) \longrightarrow A(6) \end{cases}$$

(c) $J = 0, p = 0, k = 8, B = 2$, 此时 $k > N - 1$, 跳出循环。

示例——计算过程 II

2 $J = 1$

(a) $J = 1, p = 2, k = 1, B = 2$

$$\begin{cases} A(1) + W_N^2 A(3) \longrightarrow A(1) \\ A(1) - W_N^2 A(3) \longrightarrow A(3) \end{cases}$$

(b) $J = 1, p = 2, k = 5, B = 2$

$$\begin{cases} A(5) + W_N^2 A(7) \longrightarrow A(5) \\ A(5) - W_N^2 A(7) \longrightarrow A(7) \end{cases}$$

(c) $J = 1, p = 2, k = 9, B = 2$, 此时 $k > N - 1$, 跳出循环。

倒序的概念

- ① DIT-FFT 算法流图的输出 $X(k)$ 是自然顺序
- ② 但是, 为了适应原位计算, 其输入序列不是自然顺序, 而是经过 M 次奇偶抽选后的, 这样的排序成为序列 $x(n)$ 的倒序。

顺序		倒序	
十进制 I	二进制 I'	二进制 J'	十进制 J
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

图: 顺序和倒序二进制对照表

顺序数与倒序数的不同特点

关键：若已知当前倒序数 J ，如何得到下一个倒序数。顺序数与倒序数的不同特点

- 顺序数二进制在最低位加 1，向左进位。
- 倒序数二进制在最高位加 1，向右进位。

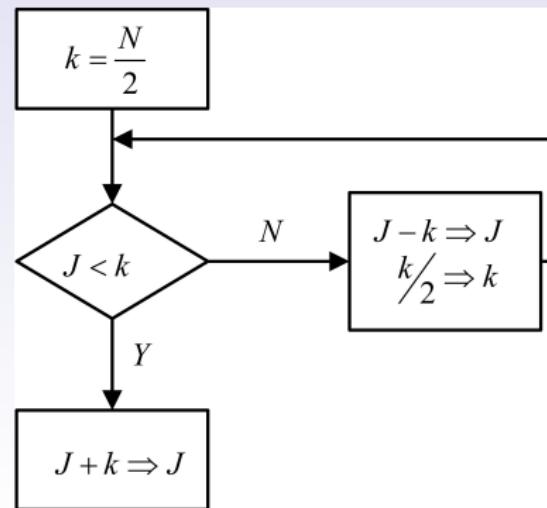
顺序		倒序	
十进制 I	二进制 I'	二进制 J'	十进制 J
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

8 点 DIT-FFT 运算流图

编程：已知当前十进制 J ，在 J' 中考察

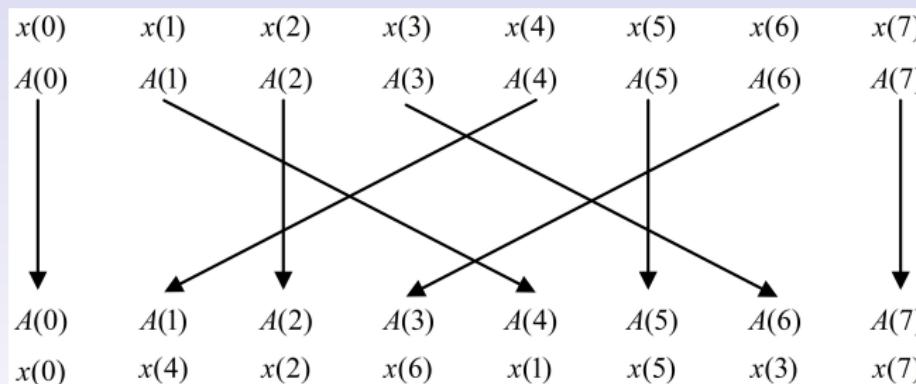
(1) 若 J' 最高位为 0，则 $J + \frac{N}{2} \rightarrow J$

(2) 若 J' 最高位为 1，则 $J - \frac{N}{2} \rightarrow J$ ，向右进一位。



对 $x(n)$ 按倒序重新排列

形成倒序后，我们已经知道倒序序列的组成。下一个问题是将原数组 A 中存放的输入序列重新按倒序排列。



- ① $x(0), x(N - 1)$ 不动。
- ② 当 $I = J$ 时，不动。
- ③ 当 $I \neq J$ 时， $A(I)$ 和 $A(J)$ 交换数据。

频域抽取法基 $2FFT$ 将长序列分解为短序列所遵循的原则

- ① 对时间前后分 (n)
- ② 对频率分奇偶 (k)

假设序列 $x(n)$ 长度为 N , 且满足 $N = 2^M$, M 为自然数。(不是的补 0)

按 n 的前后把 $x(n)$ 分解为两个 $N/2$ 长的子序列

$$X(k) = DFT[x(n)]$$

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

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

令 $n = n_1 + \frac{N}{2}$, 则有:

$$\sum_{n=\frac{N}{2}}^{N-1} x(n) W_N^{kn} = \sum_{n_1=0}^{\frac{N}{2}-1} x(n_1 + \frac{N}{2}) W_N^{k(n_1 + \frac{N}{2})}$$

(将 n_1 替换为 n)

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

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

注意: $W_N^{\frac{N}{2}k} = (W_N^{\frac{N}{2}})^k = (-1)^k$

$$\begin{aligned} X(k) &= \sum_{n=0}^{\frac{N}{2}-1} x(n) W_N^{kn} + \sum_{n=0}^{\frac{N}{2}-1} x\left(n + \frac{N}{2}\right) W_N^{\frac{N}{2}k} W_N^{kn} \\ &= \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + W_N^{\frac{N}{2}k} x\left(n + \frac{N}{2}\right) \right] W_N^{kn} \\ &= \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + (-1)^k x\left(n + \frac{N}{2}\right) \right] W_N^{kn} \quad (0 \leq k \leq N-1) \end{aligned}$$

对频率 k 奇偶分

$$\begin{cases} k = 2m \\ k = 2m + 1 \end{cases}$$

则：

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

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

而：

$$W_N^{(2m+1)n} = W_{\frac{N}{2}}^{mn} W_N^n$$

最后得到：

$$\begin{cases} X(2m) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + x\left(n + \frac{N}{2}\right) \right] W_N^{mn} & (0 \leq m \leq \frac{N}{2} - 1) \\ X(2m+1) = \sum_{n=0}^{\frac{N}{2}-1} \left[\left(x(n) - x\left(n + \frac{N}{2}\right) \right) W_N^n \right] W_N^{mn} \end{cases}$$

令，

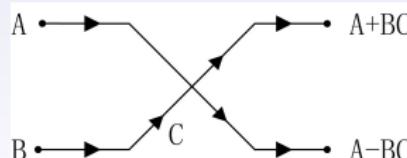
$$\begin{cases} x_1(n) = x(n) + x\left(n + \frac{N}{2}\right) & (0 \leq n \leq \frac{N}{2} - 1) \\ x_2(n) = \left[x(n) - x\left(n + \frac{N}{2}\right) \right] W_N^n \end{cases}$$

$$\text{令 } DFT[x_1(n)] = X_1(k) \quad DFT[x_2(n)] = X_2(k)$$

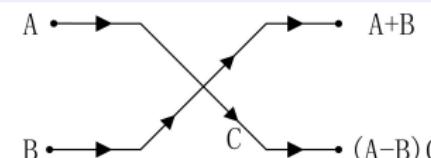
则有：

$$\begin{cases} X_1(k) = X(2k) & (0 \leq k \leq \frac{N}{2} - 1) \\ X_2(k) = X(2k + 1) \end{cases}$$

DIF-FFT 蝶形运算符号：

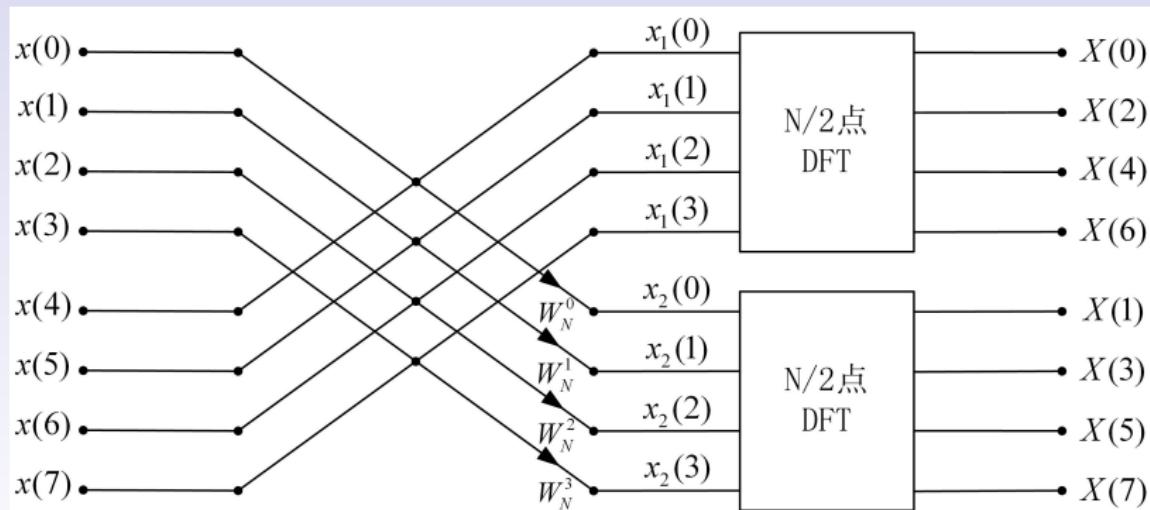


DIT-FFT蝶形运算

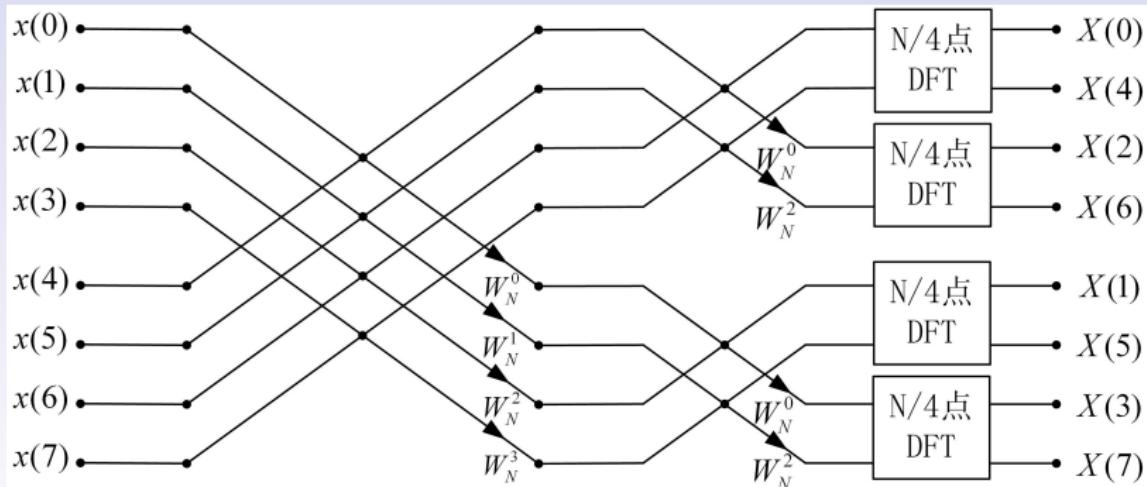


DIF-FFT蝶形运算

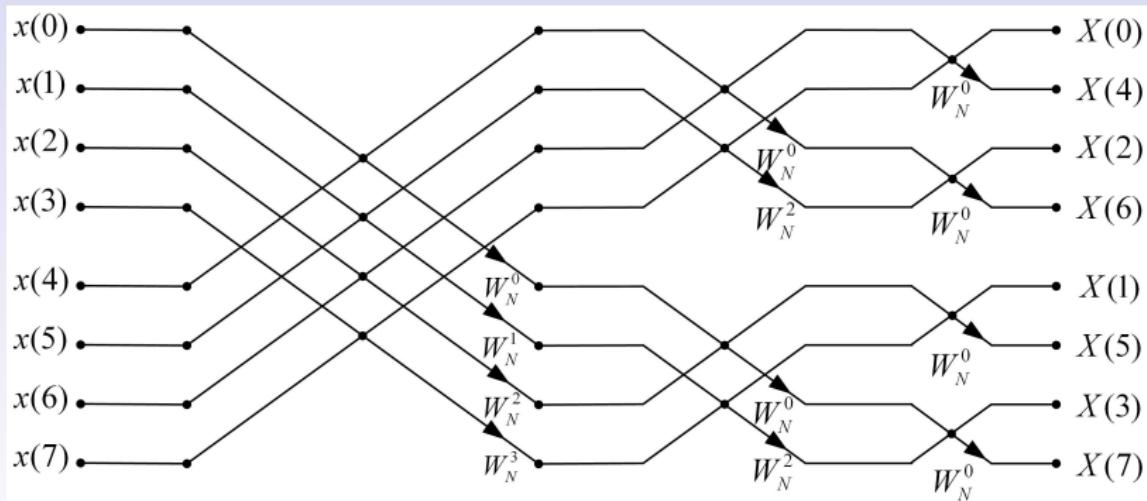
8 点 DIF-DFT 一次时域分解运算流图



8 点 DIF-DFT 二次时域分解运算流图



8 点 DIF-DFT 三次时域分解运算流图



IDFT 的快速算法

$$\begin{cases} X(k) = DFT[x(n)] &= \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad (0 \leq k \leq N-1) \\ x(n) = IDFT[X(k)] &= \frac{1}{N} \sum_{n=0}^{N-1} X(k) W_N^{-kn}, \quad (0 \leq n \leq N-1) \end{cases}$$

- 方法一：

将 $X(k)$ 作为输入, $p \rightarrow -p$, 最后的结果除以 N 。

IDFT 的快速算法

- 方法二：

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

将 $X(k)$ 取共轭，做 DFT 变换，再将结果取共轭，最后除以 N 。