数字信号处理

Digital Signal Processing

Ch3.4 快速傅里叶变换(FFT)

徐元欣，xuyx@zju.edu.cn
浙江大学信息与电子工程学院

Ch3.4.1 DFT的运算量分析

设x(n)为N点有限长序列：

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

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

两者运算量相同，下面以DFT为例分析其运算量

1. 直接计算DFT

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

一般x(n)、W_N^{nk}都是复数，X(k)也是复数。
每计算某个X(k)，需要**N**次复数乘法、**N-1**次复数加法；

∴完整的DFT共需要 **N^2**次复数乘法，**N(N-1)**次复数加法。

整个DFT共需要**$4N^2$**次实数乘法、N[2N+2(N-1)] = **2N(2N-1)**次实数加法。

综上述统计忽略如 $W_N^0=1, W_N^{N/2}=-1, W_N^{N/4}=-j$ 等不需要乘法的特例

∴DFT的运算中乘法次数，加法次数都**与N^2成正比**，当N很大时，**运算量太大**。

2. 减少运算的途径

系数W_N^{nk}有以下特性：

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

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

③可约性 $W_N^{nk} = W_{mN}^{mnk}, W_N^{nk} = W_{\frac{N}{m}}^{\frac{nk}{m}}$

④特例

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

$$W_N^{N/2} = -1$$

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

途径：
①利用这些特性，将DFT运算中的有些项进行合并
②将长序列的DFT分解成短序列的DFT
（∵N越小，运算量显著降低）

FFT的核心思想

有两者基本方法：

时间抽选法（**DIT**:**D**ecimation-**in**-**T**ime）

频率抽选法（**DIF**:**D**ecimation-**in**-**F**requency）

数字信号处理

Digital Signal Processing

Ch3.4.2 基2的时间抽选FFT
DIT-FFT

(Cookey-Tuckey算法)

徐元欣，xuyx@zju.edu.cn
浙江大学信息与电子工程学院

一、DIT-FFT算法原理

设**$N=2^L$**，L为整数（即N为2的整数幂 ∴该算法称为**基2-FFT**）

下面以DFT的计算为例进行分析

1. 用蝶形运算单元进行一次分解

将x(n)按n(n=0,1,…,N-1)的奇偶分成两组:

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

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

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

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

（n为偶数）　　（n为奇数）

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

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

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

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

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

其中令
$$\begin{cases} X_1(k)=\sum_{r=0}^{\frac{N}{2}-1}x_1(r)W_{\frac{N}{2}}^{rk}=\sum_{r=0}^{\frac{N}{2}-1}x(2r)W_{\frac{N}{2}}^{rk} \\ X_2(k)=\sum_{r=0}^{\frac{N}{2}-1}x_2(r)W_{\frac{N}{2}}^{rk}=\sum_{r=0}^{\frac{N}{2}-1}x(2r+1)W_{\frac{N}{2}}^{rk} \end{cases}$$

r=0,1,…N/2-1

k=0,1,…N-1

形式对比：

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

n,k=0,1,…N-1

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

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

如果X(k)的k取 0，1，…，$\frac{N}{2}$-1,也就是前一半的结果，则 $X_1(k)$，$X_2(k)$看做为两个N/2点的DFT。

也就是说:

（1）可以通过两个$\frac{N}{2}$- DFT来得到X(k)前半部:

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

（2）X(k)的后一半的确定

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

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

其中：

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

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

$$=X_1(k)$$

同理：

$$X_2(\frac{N}{2}+k)=X_2(k)$$

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

∴后半部分X(k)

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

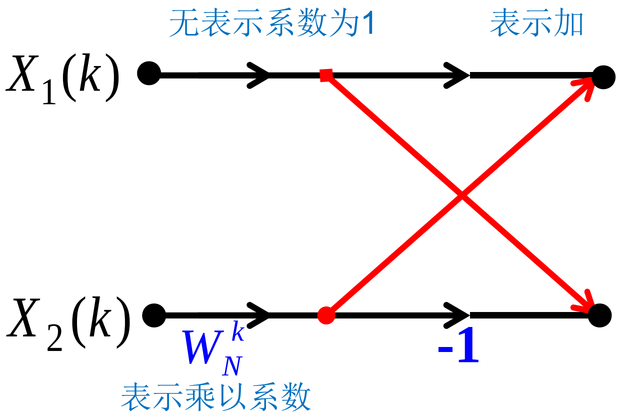
两个$\frac{N}{2}$-DFT
可以合用

$$\begin{cases} X_1(k)=\sum_{r=0}^{\frac{N}{2}-1} x_1(r)W_{\frac{N}{2}}^{rk} = \sum_{r=0}^{\frac{N}{2}-1} x(2r)W_{\frac{N}{2}}^{rk} \\ X_2(k)=\sum_{r=0}^{\frac{N}{2}-1} x_2(r)W_{\frac{N}{2}}^{rk} = \sum_{r=0}^{\frac{N}{2}-1} x(2r+1)W_{\frac{N}{2}}^{rk} \end{cases} \qquad r,k=0,1,\ldots,\frac{N}{2}-1$$

前半部分X(k)

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

前、后半部某k的运算可以用下面的**蝶形运算单元**合并表示



$X(k)=X_1(k)+W_N^k X_2(k)$ （前一半）

$X(\frac{N}{2}+k)=X_1(k)-W_N^k X_2(k)$ （后一半）

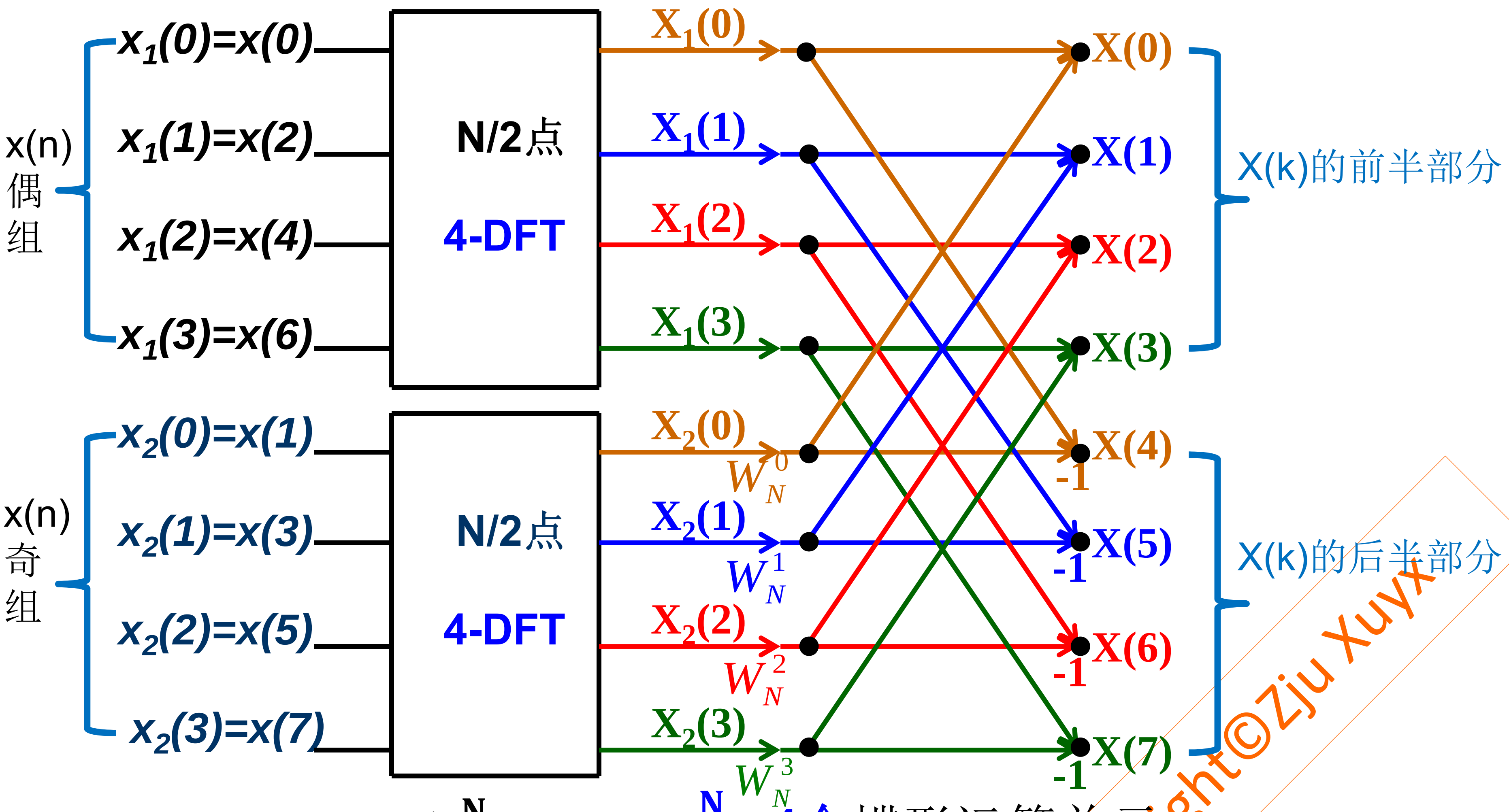
$k=0,1,\ldots,\frac{N}{2}-1$

∴只需求出 $0\sim\frac{N}{2}-1$ 内的 $X_1(k)$，$X_2(k)$ 即可求出 $0\sim N-1$ 内的 $X(k)$，

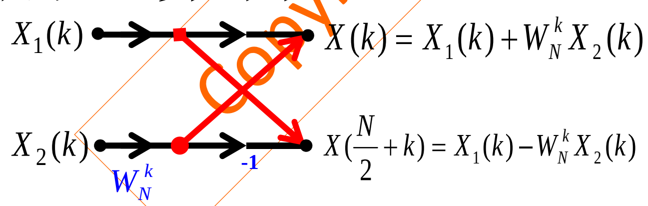
i.e. 只需**N/2**个蝶形运算单元。

每个**蝶形运算单元**需要**1**次复数乘法及**2**次复数加法

由此，设**N=8**，第一次运算分解的过程如下：



2个$\frac{N}{2}$-DFT $\frac{N}{2}$=4个蝶形运算单元



可看出，一个N点DFT分解为两个N/2点DFT

（1）如果直接计算 $\frac{N}{2}$ -DFT，则两个 $\frac{N}{2}$ -DFT所需运算量：

$2*(\frac{N}{2})^2=\frac{N^2}{2}$ 次复数乘法

$2*(\frac{N}{2})(\frac{N}{2}-1)=N(\frac{N}{2}-1)$ 次复数加法

（2）另，有$\frac{N}{2}$个蝶形运算，还需：

$\frac{N}{2}*1=\frac{N}{2}$ 次复数乘法

$\frac{N}{2}*2=N$ 次复数加法

∴该方法一次分解过程共需运算量：

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

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

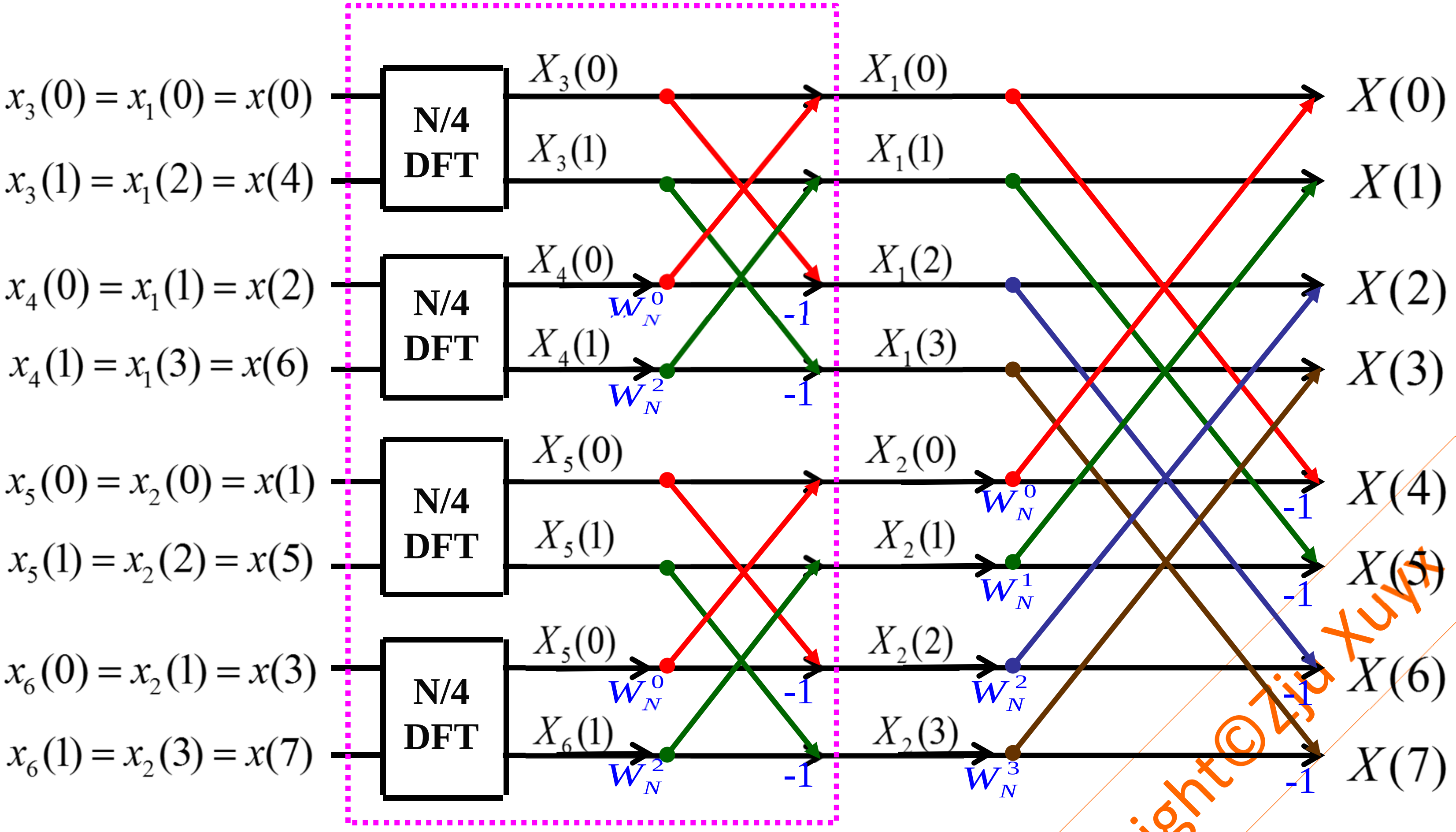
由前可知：直接计算N-DFT共需要 **N^2**次复数乘法，**N(N-1)**次复数加法

一次分解法比直接运算N点的DFT的运算量差不多减少一半

由此可以将两个N/2点DFT进一步分解

2、将每个N/2点的序列进一步分为两个N/4点组合

则**N=8**点DFT可进一步分解为4个N/4=2点DFT：



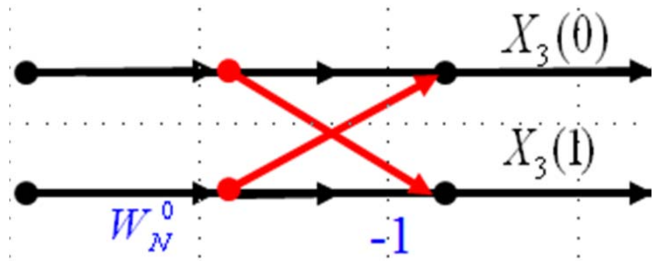
4个2-DFT **N/2个**蝶形运算单元 **N/2个**蝶形运算单元

其中将系数统一为：$W_{\frac{N}{2}}^{k}=W_N^{2k}$

其中，N/4=2点DFT同样可直接用蝶形运算单元实现比如：

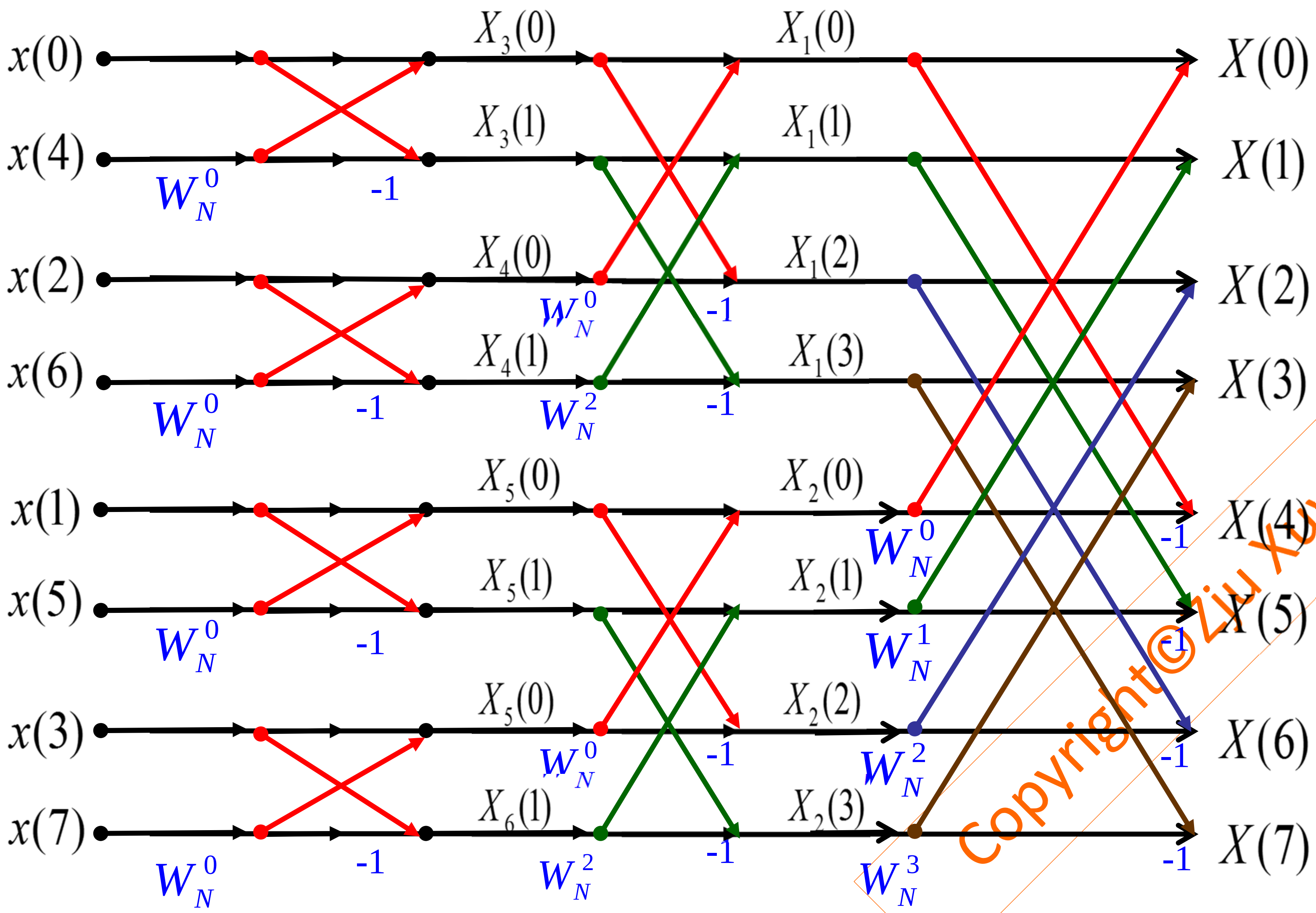
$$x_3(0) = x_1(0) = x(0)$$

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



为了统一运算结构，2点DFT仍采用系数W_N^0表示

因此，**N=8**点的DIT-FFT运算流图如下：



二、运算量

由前面流图可知，对于$N=2^L$的DIT-FFT总体结构为：

- 有L级蝶形运算
- 每级蝶形运算都由N/2个蝶形运算单元组成

对于$N=2^L$：

- 有L级蝶形运算
- 每级蝶形运算都由N/2个蝶形运算单元组成
- 每个蝶形运算单元需1次复乘，2次复加

∴FFT的运算量为

复乘： $m_F = L*\frac{N}{2}*1 = \frac{N}{2}\log_2 N$ 次

复加： $a_F = L*\frac{N}{2}*2 = N\log_2 N$ 次

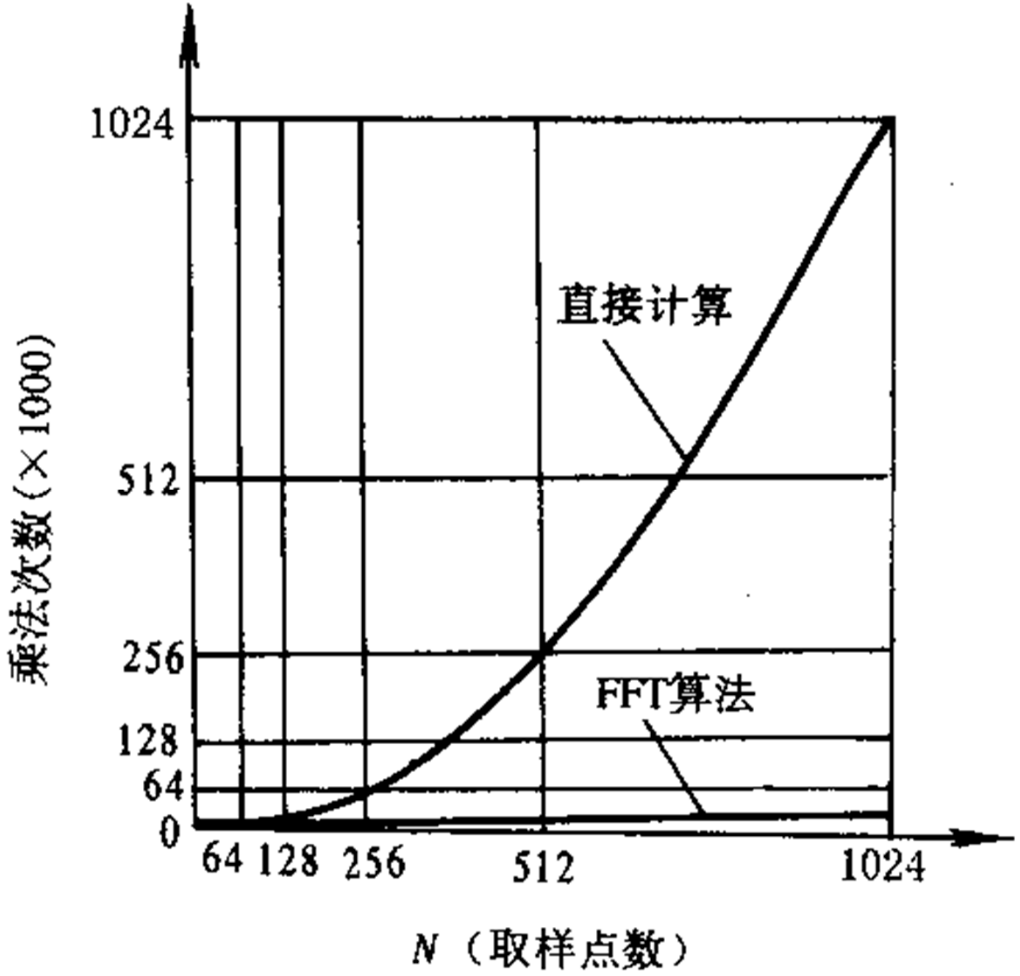
注：实际运算量数字稍有不同，如特例$W_N^0=1$，$W_N^{N/4}=-j$，不需乘法，当N很大时，这些相对比例很少，所以运算量就不考虑这些特例。

以乘法运算量为参照进行比较（∵乘法运算时间/硬件资源>>加法）

直接计算DFT与FFT算法的计算量之比为：

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

FFT算法与直接计算DFT所需乘法次数的比较曲线



FFT算法与直接算法的运算量比较

N	$N^2/(\frac{N}{2}\log_2 N)$
2	4.0
4	4.0
8	5.4
16	8.0
32	12.8
64	21.4
128	36.6
256	64.0
512	113.8
1024	204.8
2048	372.4

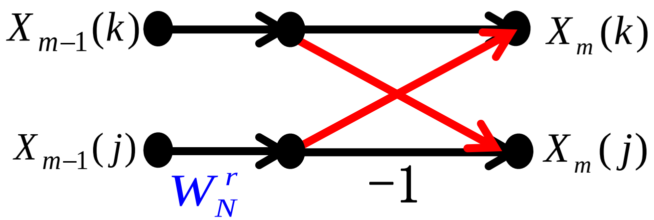
当N越大，FFT的优点越突出

三、DIT-FFT算法的特点

由以上**特点**，可以画出任意**N=2^L**点的**DIT-FFT**算法的流图，步骤如下：

① 输入**倒位序**，输出自然序

②总体结构：共有**L列**，每列有**N/2个**蝶形运算单元



③蝶形两节点**距离**（j-k）=**2^{m-1}**

W_N^r：**r**为行号**k左移(L-m)**，右补0

m：第m列，1,2,…,L

k：第1个节点所在行号，0,1,…,N-1

画信号流图注意事项：

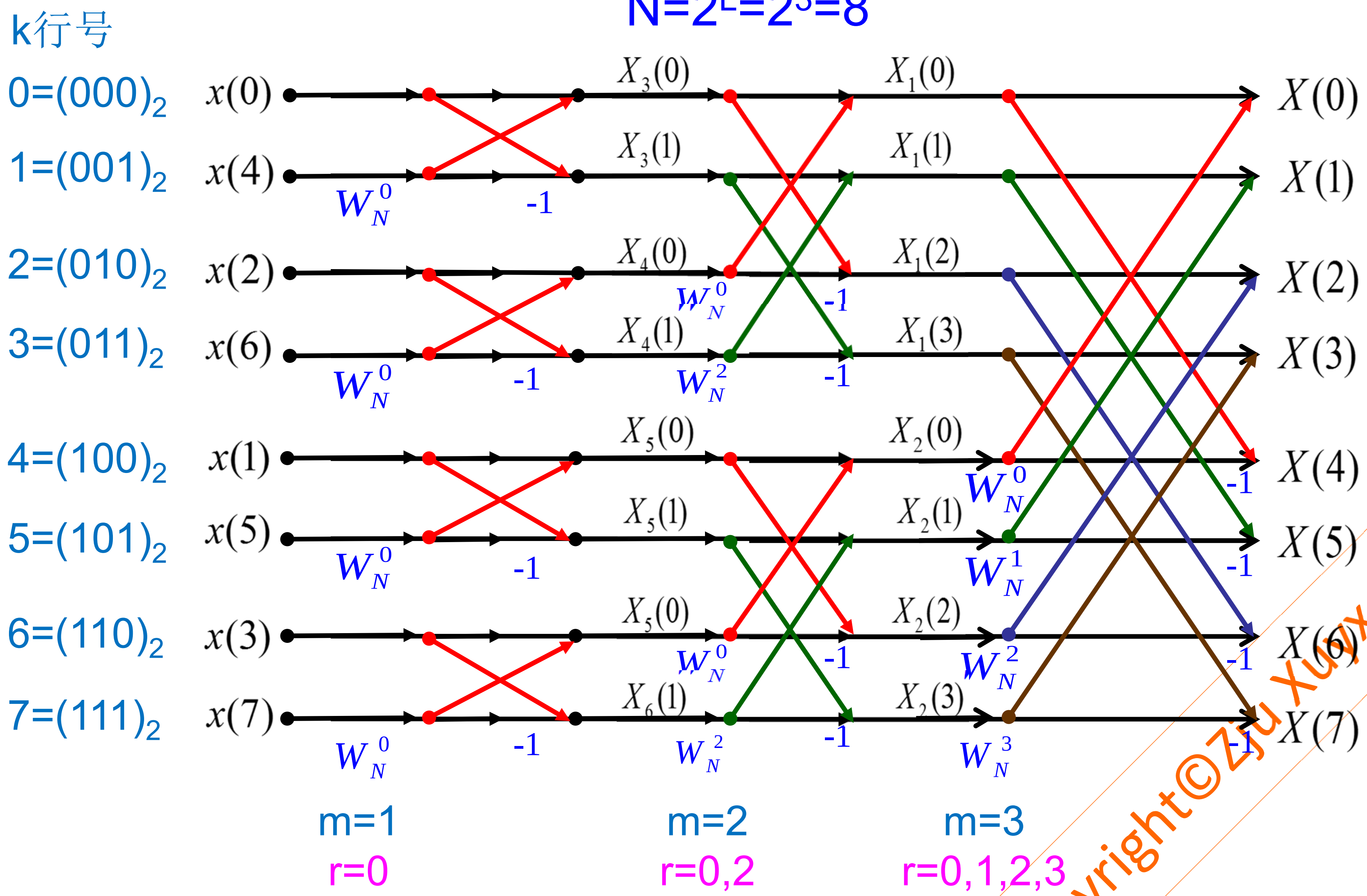
①节点

②箭头

③乘数因子

④相乘，“-1”

$N=2^L=2^3=8$



W_N^r的r：①将第一个节点$X_m(k)$的行号k用3位二进制表示

②再将该二进制数左移3-m位（右边补0），就得到r的二进制数

数字信号处理

Digital Signal Processing

Ch3.4.3 基2的频率抽选FFT
DIF-FFT

(Sande-Tukey算法)

徐元欣，xuyx@zju.edu.cn
浙江大学信息与电子工程学院

一、算法原理

设序列点数**N=2^L**，L为正整数。**DIT**是把输入**x(n)**按其顺序的**奇偶**进行分解，**DIF**则是把输出**X(k)**按其顺序的**奇偶**进行分解。

1、将x(n)按n顺序分成前后两半

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

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

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

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

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

$$\because W_N^{\frac{N}{2}}=-1,\therefore W_N^{\frac{Nk}{2}}=(-1)^k$$

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

k=0,1,…,N-1

将X(k)按**k的奇偶**分为两部分

k偶数

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

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

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

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

k奇数

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

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

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

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

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

则

X(k)
偶数

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

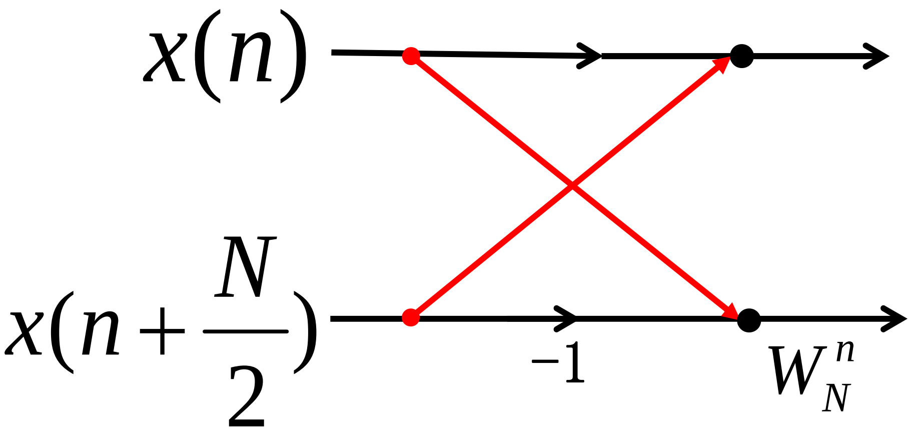
$r = 0,1\cdots\cdots\frac{N}{2}-1$

X(k)
奇数

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

两个$\frac{N}{2}$点的DFT

DIF的蝶形运算单元：



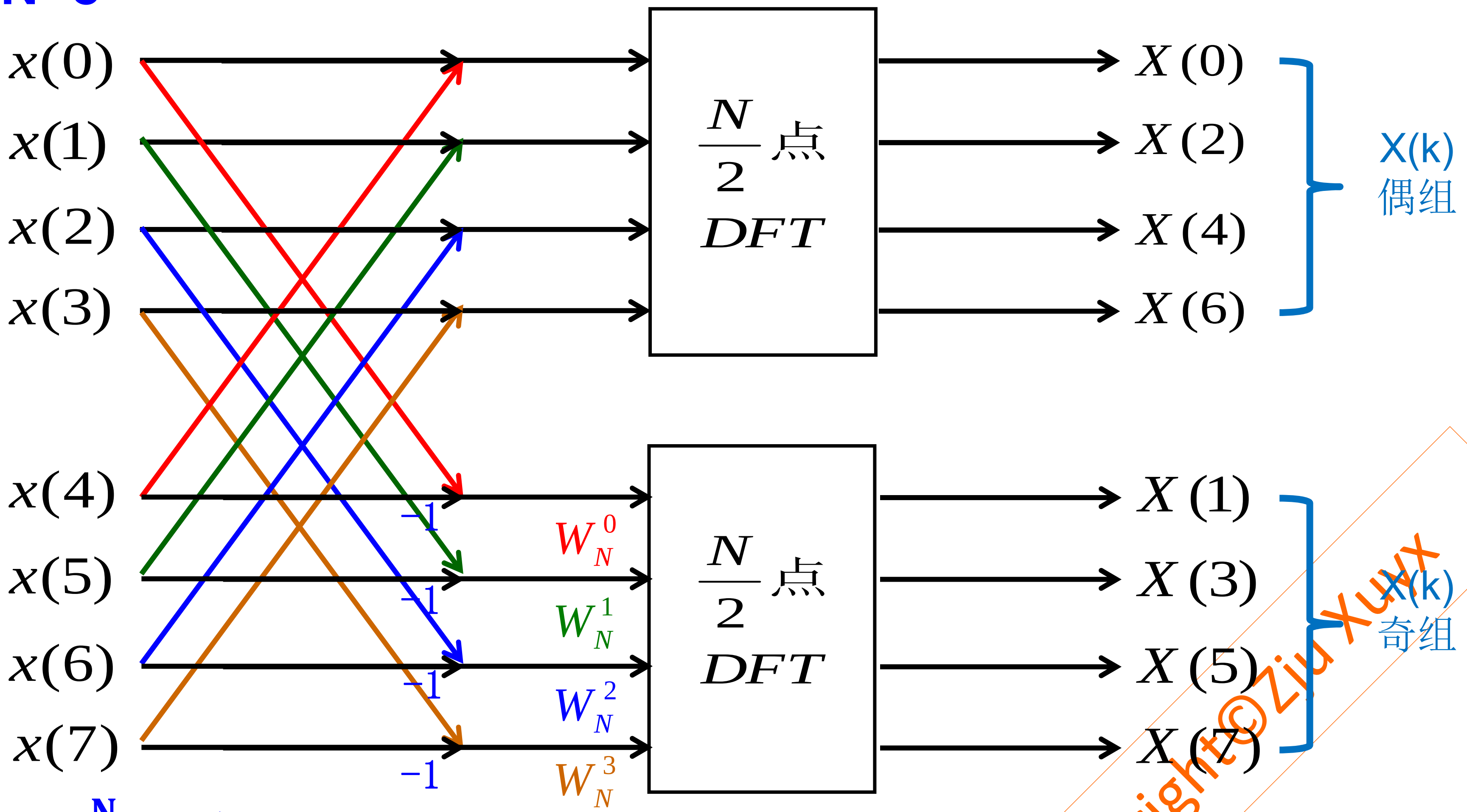
$$x(n)+x(n+\frac{N}{2})$$

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

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

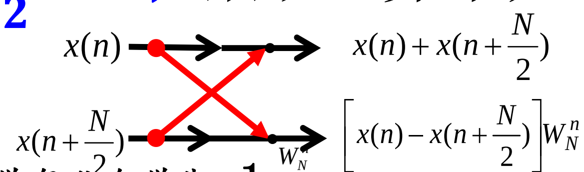
由此，N点DFT按k的奇偶分解为两个$\frac{N}{2}$点的DFT

N=8



$\frac{\mathbf{N}}{\mathbf{2}}$ **=4个**蝶形运算单元

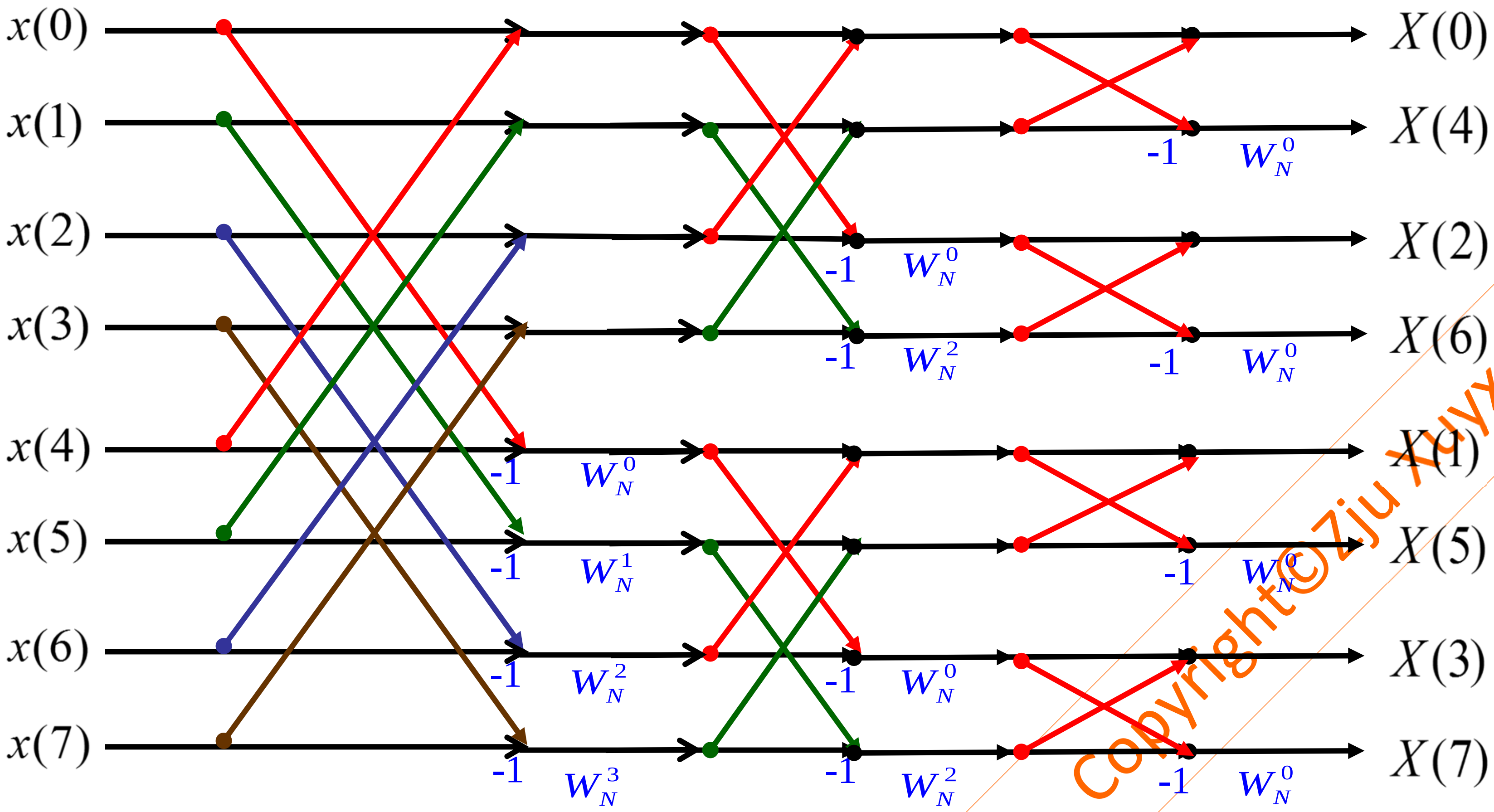
2个$\frac{\mathbf{N}}{\mathbf{2}}$ -DFT



由此，将$\frac{N}{2}$点DFT进一步分解为两个$\frac{N}{4}$点DFT…一直到**第L次**。

注：第L次的2点DFT仍用系数W_N^0 蝶形运算以统一运算结构。

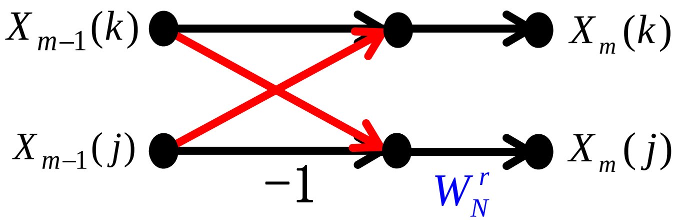
N=8



∴由以上**特点**，可以画出任意**N=2^L**点的**DIF-FFT**算法的流图，步骤如下：

①输入自然序，输出倒位序

②总体结构：共有**L列**，每列有**N/2个**蝶形运算单元



③蝶形两节点距离（j-k）=**2^{L-m}**

W_N^r：**r**为行号**k左移(m-1)**，右补0

m：第m列，1,2,…,L

k：第1个节点所在行号，0,1,…,N-1

数字信号处理

Digital Signal Processing

Ch3.4.5 IDFT的快速计算方法（IFFT）

徐元欣，xuyx@zju.edu.cn
浙江大学信息与电子工程学院

1、由FFT $\xrightarrow{\text{改动}}$ IFFT

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

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

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

由上可看出：
前面的DIT、DIF的FFT也可适用于IDFT，只要将DFT的每个运算系数W_N^{nk}换成 W_N^{-nk}，最后系数乘以$\frac{1}{N}$ 。

见书p122的图3.41

2、由FFT $\xrightarrow{\text{直接}}$ IFFT

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

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

步骤:

①将X(k)取共轭

②利用FFT程序计算

③将结果取共轭

④再乘以 $\frac{1}{N}$，得到x(n)

数字信号处理

Digital Signal Processing

Ch3.5.1 线性卷积的快速计算

徐元欣，xuyx@zju.edu.cn
浙江大学信息与电子工程学院

设x(n)为**L**点，h(n)为**M**点(FIR滤波器)

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

y(n)为L+M-1点

1. 直接计算线性卷积

该线性卷积共需 $m_d=LM$ 次乘法(复乘)。

特别的对于线性相位的FIR滤波器，有：

$$h(n)=\pm h(M-1-n)$$

Ch5将讲

由此，线性相位FIR滤波器所需乘法：$m_d=\dfrac{LM}{2}$

2. 采用圆周卷积代替线性卷积 → FFT方法

由Ch3.3.3节，p96可知：

为了不产生混叠，将**L**点的x(n)、**M**点的h(n)各自补0，都成为**N ≥ L+M-1**点。

$$x(n)=\begin{cases} x(n) & 0\le n\le L-1 \\ 0 & L\le n\le N-1 \end{cases}$$

$$h(n)=\begin{cases} h(n) & 0\le n\le M-1 \\ 0 & M\le n\le N-1 \end{cases}$$

则有：

$$y(n)=x(n)*h(n)=x(n)\,\text{Ⓝ}\,h(n)$$

y(n)为L+M-1点

采用DFT/IDFT来运算，更进一步采用FFT/IFFT快速运算

∴对N点圆周卷积利用FFT来计算，分以下4步：

① $H(k)=FFT[h(n)]$ ，N点

② $X(k)=FFT[x(n)]$ ，N点

③ $Y(k)=X(k)H(k)$

④ $y(n)=IFFT[Y(k)]$ ，N点

该快速卷积方法共需**复乘**： $m_F=3\times\frac{N}{2}\log_2 N+N=N(1+\frac{3}{2}\log_2 N)$

因此，前面第1方法直接线性卷积(线性相位FIR)与本快速卷积方法的乘法运算量之比：

$$k_m=\frac{m_d}{m_F}=\frac{\frac{LM}{2}}{N(1+\frac{3}{2}\log_2 N)}=\frac{LM}{2(M+L-1)(1+\frac{3}{2}\log_2(M+L-1))}$$

分两种情况：

(1). x(n)与h(n)点数差不多

设$M=L$，则$N=2M-1\approx 2M$

$$\therefore k_m=\frac{M}{10+6\log_2 M}$$

参见下表可知：M超过64点，M越大，快速卷积方法越有优势

M=L	8	32	64	128	256	512	1024	2048	4096
k_m	0.256	0.80	1.39	2.46	4.41	8	14.62	26.15	49.95

(2). x(n)点数很多

$$L \gg M, 则 N = L + M - 1 \approx L$$

$$\therefore k_m = \frac{M}{2 + 3\log_2 L}$$

$$\therefore L 太大时，k_m 会下降$$

此时可采用分段卷积(或分段过滤)方法：

1）重叠相加法

2）重叠保留法

参见“**Ch3.3.4**节 **1.**线性卷积的逐段计算方法”

1) 重叠相加法

将x(n)分解成多段$x_i(n)$,每段为L点: **L≈M**，即两者同一量级

其中第i段

$$x_i(n) = \begin{cases} x(n) & iL \le n \le (i+1)L-1 \\ 0 & other \end{cases}$$

$$i = 0,1,2,\cdots$$

有

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

因此

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

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

对于每段 $y_i(n)=x_i(n)*h(n)$ 可用前面的快速卷积来计算：

将$x_i(n)$的n序号改为从0开始，再将$x_i(n),h(n)$都补零至**N**点：

$$N=2^m \geq L+M-1$$

采用快速卷积计算得到每段结果：$y_i(n)=x_i(n) \text{Ⓝ} h(n)$

每段结果为**N**点(其中后面N-(L+M-1)点为0)，将其起点位置重新调回到**iL**处进行拼接：

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

最后的输出为每段快速卷积的结果相拼接而成：

前一段的后**(M-1)**点和后一段的前**M-1**个点相重叠，这就是重叠相加法。

∴采用快速卷积的**重叠相加**方法步骤为:

① 分段，每段$x_i(n)$为L点: **L≈M**， n序号改为从0开始

② $H(k)=FFT[h(n)]$ ，N点

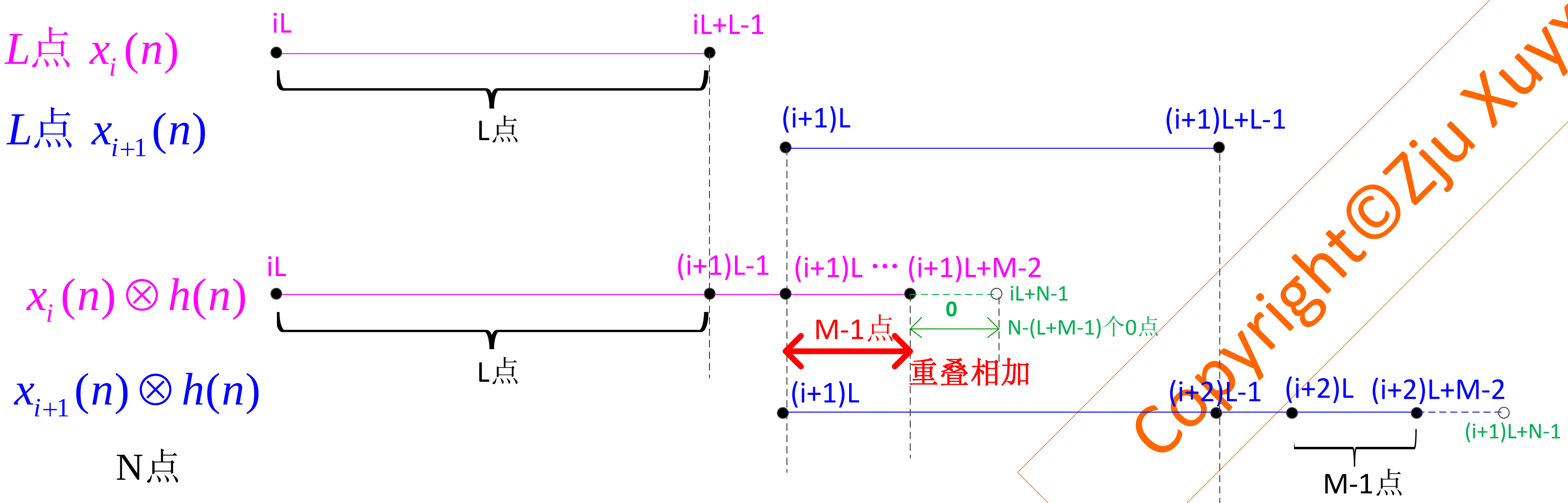
③ $X_i(k)=FFT[x_i(n)]$ ，N点

$$N=2^m \ge L+M-1$$

④ $Y_i(k)=X_i(k)H(k)$

⑤ $y_i(n)=IFFT[Y_i(k)]$ ， N点

⑥ 每段结果起始位置重新调回到**iL**处进行拼接 $y(n)=\sum_{i=0}^{\infty} y_i(n)$



2) 重叠保留法

先将按每段**L**=N-M+1 点对x(n)进行分段，然后在每一段前面补上前一段保留下来的**(M-1)**点输入序列的值。因此每段成为**N'=L+M-1**点：

$$x_i(n)=\begin{cases} x(n) & iL-M+1\le n\le (i+1)L-1 \\ 0 & other \end{cases} \qquad i=1,2,3,\cdots$$

对于第一段则为：

$$x_0(n)=\begin{cases} 0, & -M+1\le n\le -1 \\ x(n) & 0\le n\le L-1 \\ 0 & other \end{cases}$$

也就是说每段长度为**N'=L+M-1**点，相邻两段之间存在**M-1**点重叠

∴采用快速卷积的**重叠保留**方法步骤为：

n序号改为从0开始

① 分段，每段$x_i(n)$为**L**点，前面再补上前一段保留下来的**(M-1)**点，尾部再补0到N点

② $H(k)=FFT[h(n)]$ ，N点 　　$h(n)$尾部补0到N点

③ $X_i(k)=FFT[x_i(n)]$ ，N点

④ $Y_i(k)=X_i(k)H(k)$ 　　$N=2^m\geq L+M-1$

⑤ $y_i(n)=IFFT[Y_i(k)]$ ，N点

⑥ 将每段结果有重叠的前面**M-1**点及后面**N-(L+M-1)**点，余下没重叠的**L**点起始位置重新调回到**iL**处进行拼接

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