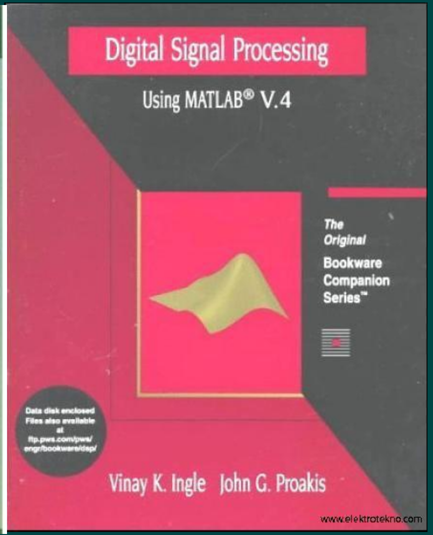
数字信号处理

任课教师：田春娜（博士）
单位：西安电子科技大学电子工程学院
Email: chnatian@xidian.edu.cn
chnatian@gmail.com

参考书籍



•高新波, 阔永红, 田春娜. 数字信号处理. 高等教育出版社, 2014.

•史林, 赵树杰. 数字信号处理. 科学出版社. 2007.

•Alan V. Oppenheim, Ronald W. Schafer. Discrete-Time Signal Processing. 电子工业出版社, 2011.

•高西全, 丁玉美. 数字信号处理及其习题解答. 西电出版社, 2008.

•Vinay K.Ingle, John G. Proakis. Digital Signal Processing Using MATLAB®. Northeastern University, 1996.

引言

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

运算量	复数乘法	复数加法
一个$X(k)$	N	$N-1$
N个$X(k)$ (N点DFT)	N^2	$N(N-1)$

	实数乘法	实数加法	
一次复乘	4	2	$(a+jb)(c+jd)=(ac-bd)+j(ad+cb)$
一次复加		2	$(a+jb)+(c+jd)=(a+c)+j(b+d)$

	实数乘法	实数加法
一个$X(k)$	$4N$	$2N+2(N-1)=2(2N-1)$
N个$X(k)$ (N点DFT)	$4N^2$	$2N(2N-1)$

引言

当N较大时，计算量太大，所以在**快速傅里叶变换(Fast Fourier Transform, FFT)**出现以前，直接用DFT算法进行谱分析和信号的实时处理是不切实际的

直到1965年库利(J. W. Cooley)和图基(J .W. Tukey)发表的“An algorithm for the machine calculation of complex Fourier series. Math. Comp., Vol. 19 (April 1965), pp. 297–301”以后，研究人员相继改进了DFT快速算法，情况才发生了根本的改变

引言

降低运算量的途径

把N点DFT分解为几个较短的DFT，可使乘法次数大大减少，另外，利用旋转因子$W^m{}_N$的**周期性**、**对称性**和**可约性**来减少DFT的运算次数

$$W_N^{m+lN}=e^{-j\frac{2\pi}{N}(m+lN)}=e^{-j\frac{2\pi}{N}m}=W_N^m$$ ← **周期性**

$$W_N^{-m}=W_N^{N-m} \quad [W_N^{N-m}]^*=W_N^m \quad W_N^{m+\frac{N}{2}}=-W_N^m$$ ← **对称性**

$$W_N^m=W_{N/n}^{m/n}, \quad N/n,m/n\text{为整数}$$ ← **可约性**

利用这些性质提出：基2FFT、基4FFT、分裂基FFT、DHT等快速傅立叶变换算法

引言

- 此外，还有复合数快速傅里叶变换算法，使用于N为复合数的情况，即N可表示为若干因子之乘积

$$N = r_1, r_2, \cdots, r_{L_0}$$

基2算法是这种方法的一个特例

- 如果序列的长度N不是2^M，且N又是素数，如果求准确的N点DFT，可用线性调频Z变换(Chirp-Z变换)方法求FFT.

第五章 快速傅里叶变换(FFT)

- 5.1 基2FFT算法
  - 5.1.1 时域抽取基2FFT算法
  - 5.1.2 频域抽取基2FFT算法
- 5.2 IDFT的快速算法
- 5.3 实序列的FFT算法

5.1 基2FFT算法

依次分解为短序列并结合旋转因子特性计算

- 基2时间抽取(Decimation in time) DIT-FFT

$$x(n) \rightarrow \begin{cases} x(2r) \\ x(2r+1) \end{cases} \quad r = 0,1,\cdots \frac{N}{2}-1$$

奇偶分组

- 基2频率抽取(Decimation in frequency) DIF-FFT

$$X(k) \rightarrow \begin{cases} X(2m) \\ X(2m+1) \end{cases} \quad m = 0,1,\cdots \frac{N}{2}-1$$

第五章 快速傅里叶变换(FFT)

- 5.1 基2FFT算法
  - 5.1.1 时域抽取基2FFT算法
  - 5.1.2 频域抽取基2FFT算法
- 5.2 IDFT的快速算法
- 5.3 实序列的FFT算法

5.1.1 时域抽取基2FFT算法

1. 算法原理

设序列点数 $N = 2^L$，L为正整数。 若不满足，则补零，

使 N 为2的整数幂的FFT算法称为基-2 FFT算法

将序列 $x(n)$ 按n的奇偶分两组

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

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

$$r = 0,1,\cdots\cdots, N/2-1$$

$$X(k) = \mathrm{DFT}\left[x(n)\right] = \sum_{n=0}^{N-1} x(n) W_N^{nk} = \sum_{n=even} x(n) W_N^{kn} + \sum_{n=odd} x(n) W_N^{kn}$$

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

$$\boxed{\begin{gathered} x(2r) = x_1(r) \\ x(2r+1) = x_2(r) \\ r = 0,1,\cdots\cdots,N/2-1 \end{gathered}}$$

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

旋转因子可约性

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

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

$$= X_1(k) + W_N^k X_2(k), \quad k = 0,1,\cdots,N/2-1$$

$$X_1(k) = \sum_{r=0}^{N/2-1} x_1(r) W_{N/2}^{kr} = \mathrm{DFT}[x_1(r)]$$

$$X_2(k) = \sum_{r=0}^{N/2-1} x_2(r) W_{N/2}^{kr} = \mathrm{DFT}[x_2(r)]$$

前$N/2$点的DFT

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

$$0 \le k \le N-1$$

$$N/2 \le k \le N \qquad 0 \le k \le N/2-1$$

？？？

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

$$0\le k\le N-1$$

$$N/2\le k\le N$$

？？？

$$0\le k\le N/2-1$$

利用周期性求 $X(k)$ 的后半部分

$\because X_1(k), X_2(k)$ 是以 $N/2$ 为周期的

$$\therefore X_1(k+\frac{N}{2})=X_1(k) \qquad X_2(k+\frac{N}{2})=X_2(k)$$

又 $W_N^{k+\frac{N}{2}}=W_N^{N/2}W_N^k=-W_N^k$ 后半个序列的求法如下：

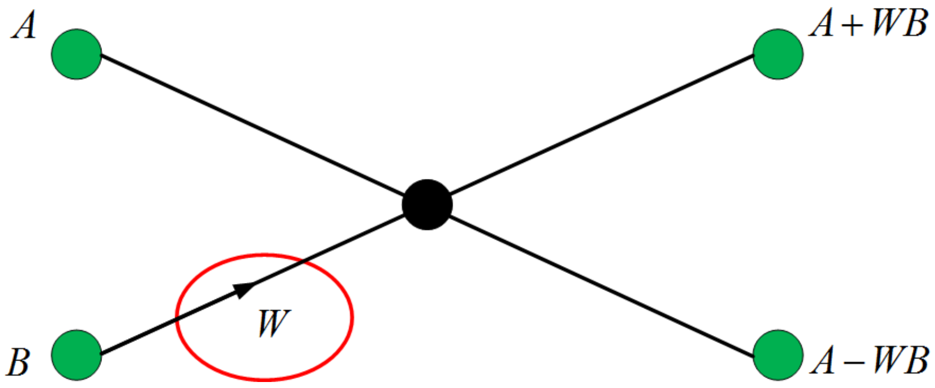
$$X(k+N/2)=X_1(k+N/2)+W_N^{(k+N/2)}X_2(k+N/2)$$

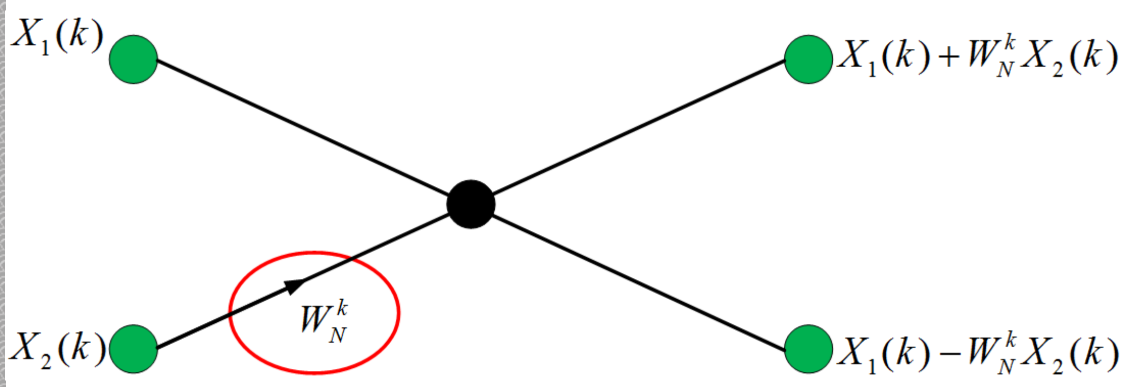
$$X(k+N/2)=X_1(k)-W_N^k X_2(k) \qquad 0\le k\le N/2-1$$

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

隐含周期性

旋转因子对称性

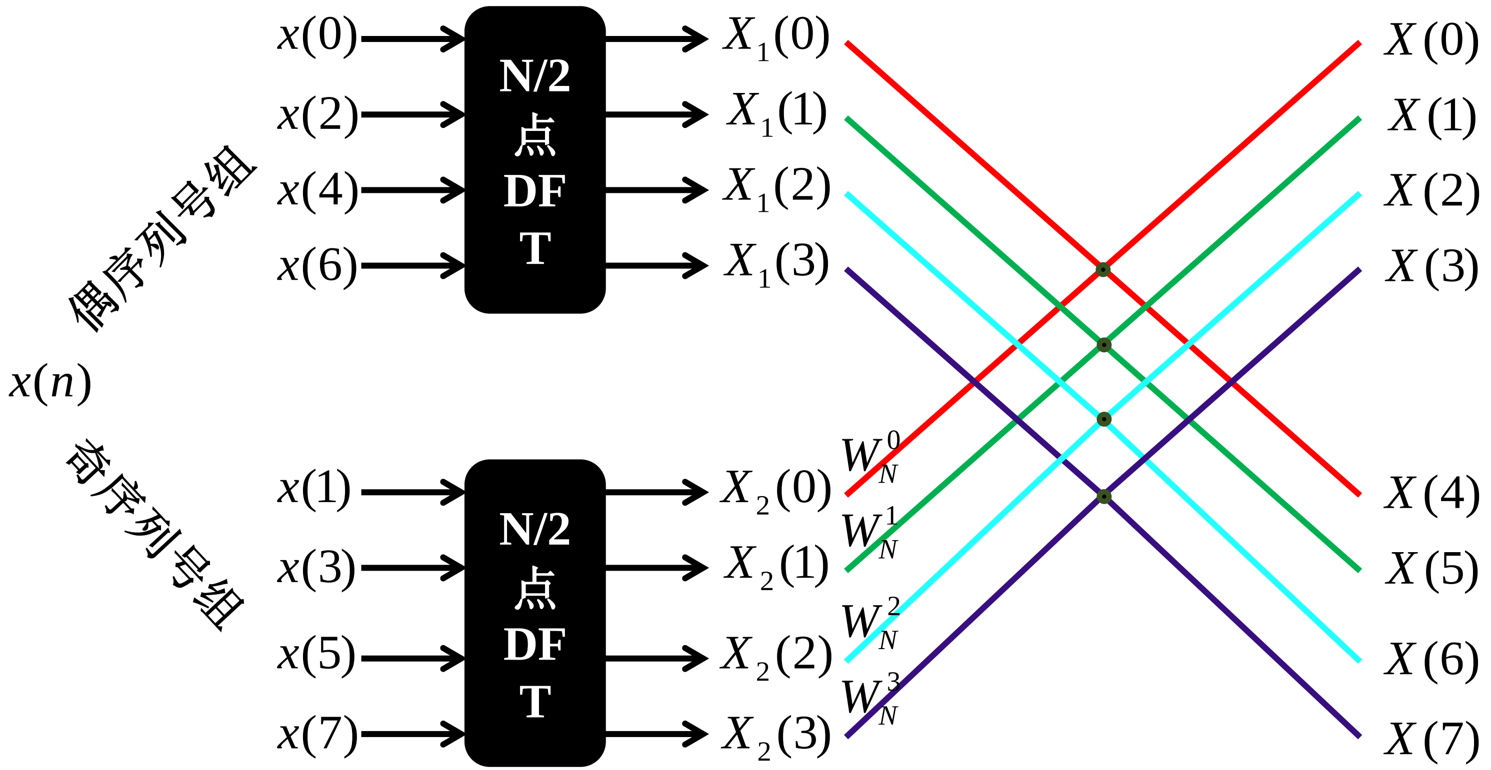




蝶形单元需要
一次复数乘法
两次复数加法

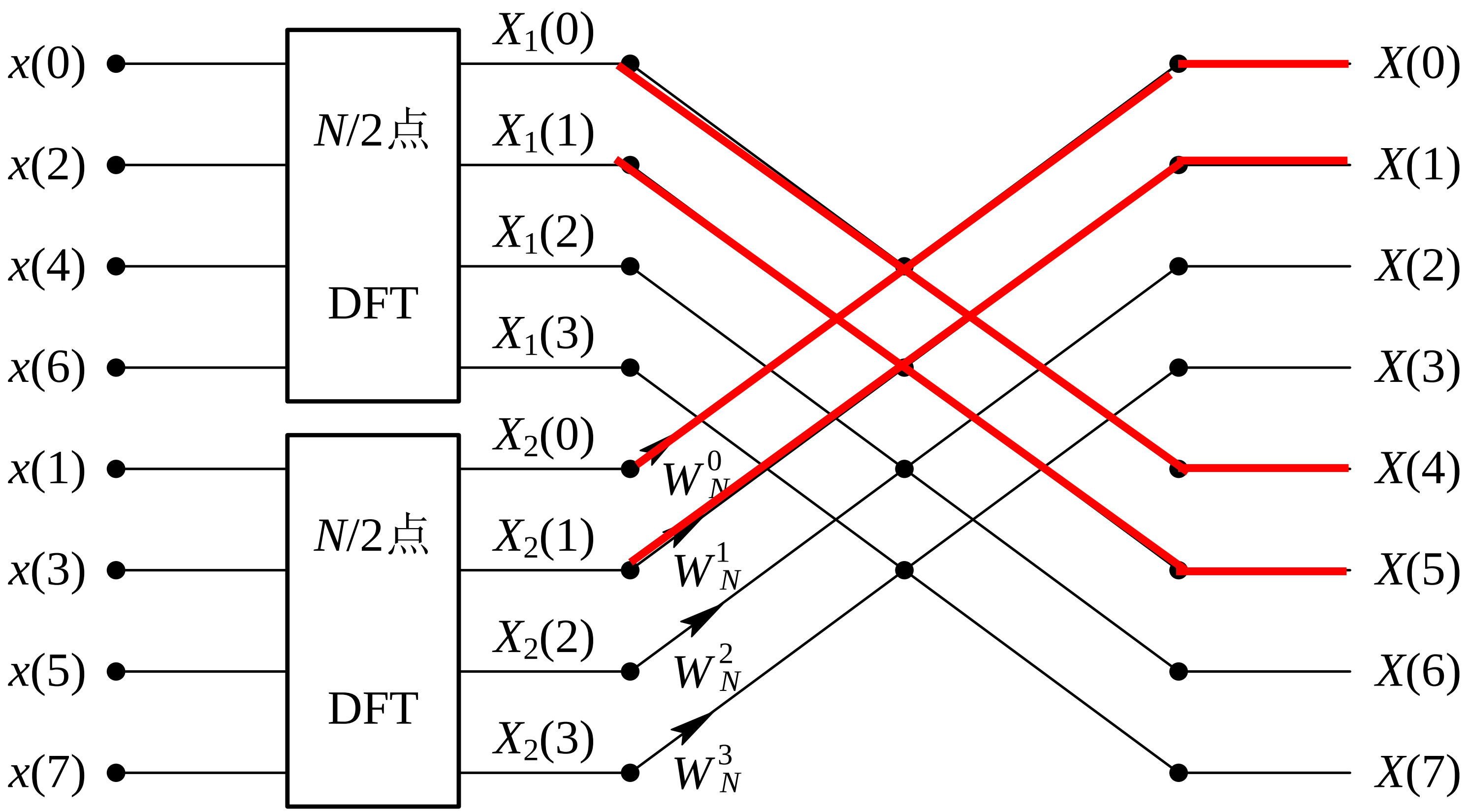
$$X(k)=X_1(k)+W_N^k X_2(k)$$

$$X(k+N/2)=X_1(k)-W_N^k X_2(k) \qquad 0\le k\le N/2-1$$



$$X(k)=X_1(k)+W_N^k X_2(k) \qquad 0\le k\le N/2-1$$

$$X(k+N/2)=X_1(k)-W_N^k X_2(k)$$



N点DFT的一次时域抽取分解图(N=8)

$N/2$仍为偶数，进一步分解：$N/2 \rightarrow N/4$

$$\begin{cases} x_1(2l) = x_3(l) \\ x_1(2l+1) = x_4(l) \end{cases} \qquad l = 0,1,\cdots\cdots, \quad N/4-1$$

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

其中

$$X_3(k) = DFT[x_3(l)] = DFT[x_1(2l)]$$

$$X_3(k) = DFT[x_4(l)] = DFT[x_1(2l+1)]$$

$$l = 0,1,\cdots\cdots, \quad N/4-1$$

同理

$$\begin{cases} X_2(k) = X_5(k) + W_{N/2}^{k} X_6(k) \\ X_2(k+\dfrac{N}{4}) = X_5(k) - W_{N/2}^{k} X_6(k) \end{cases}$$

$$k = 0,1,\cdots\cdots,\frac{N}{4}-1$$

其中

$$X_5(k) = DFT[x_5(l)] = DFT[x_2(2l)]$$

$$X_6(k) = DFT[x_6(l)] = DFT[x_2(2l+1)]$$

$$l = 0,1,\cdots\cdots,\quad N/4-1$$

5.1.1 时域抽取基2FFT算法

◆二次分解

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

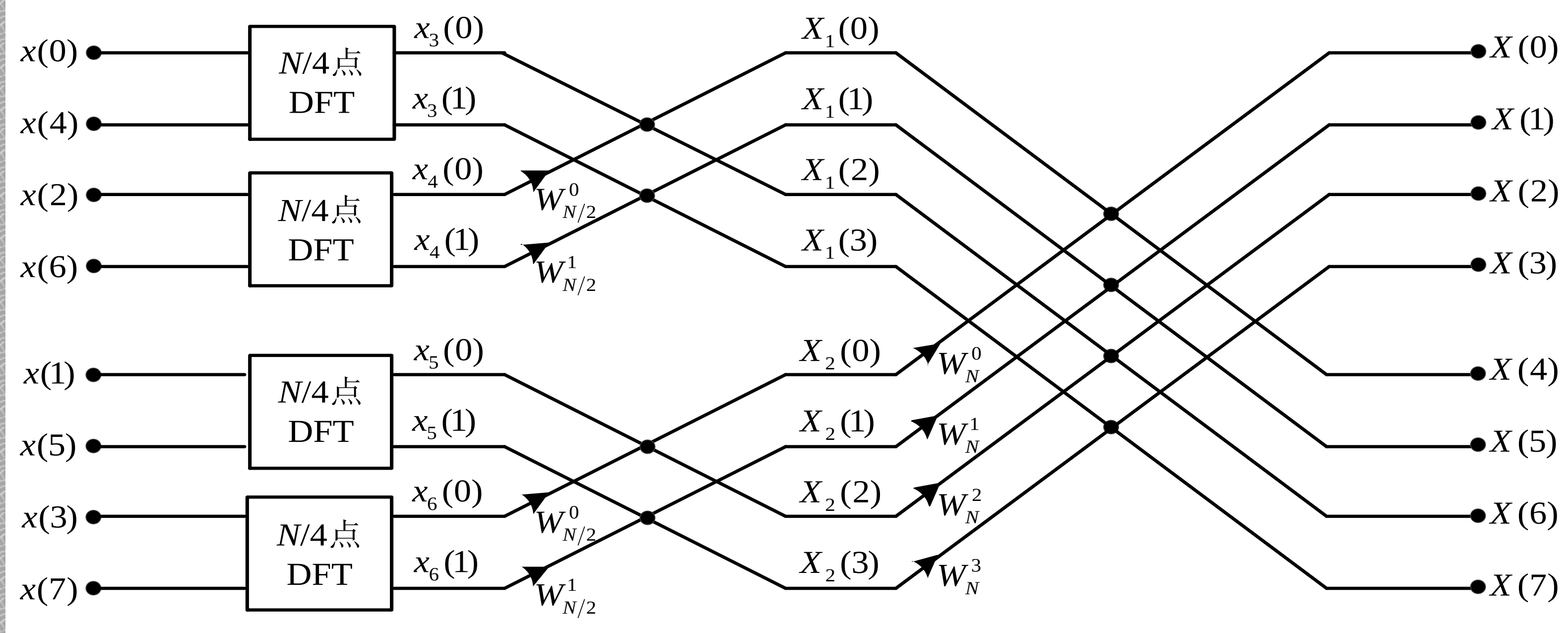
$$=\sum_{l=0}^{N/4-1}x_3(l)W_{N/4}^{kl}+W_{N/2}^{k}\sum_{l=0}^{N/4-1}x_4(l)W_{N/4}^{kl}$$

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

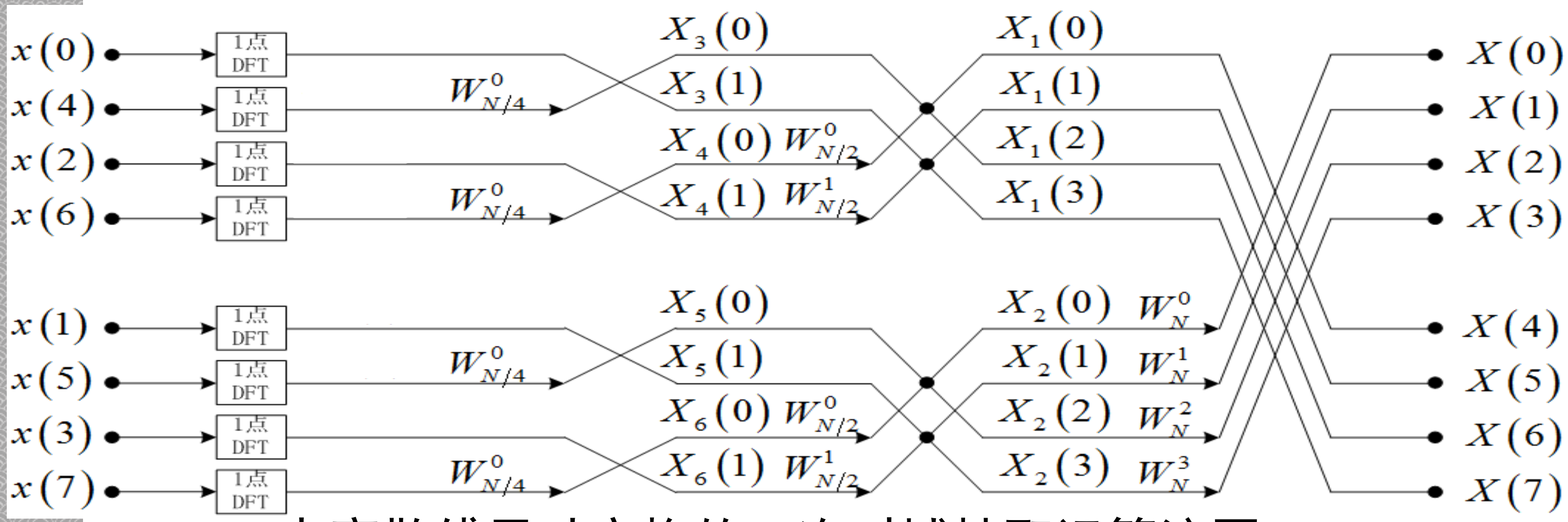
$$\begin{cases}
X_1(k)=X_3(k)+W_{N/2}^{k}X_4(k), & 0\le k\le N/4-1\\
X_1(k+N/4)=X_3(k)-W_{N/2}^{k}X_4(k), & 0\le k\le N/4-1\\
X_2(k)=X_5(k)+W_{N/2}^{k}X_6(k), & 0\le k\le N/4-1\\
X_2(k+N/4)=X_5(k)-W_{N/2}^{k}X_6(k), & 0\le k\le N/4-1
\end{cases}$$

$$\begin{cases} x_3(l) = x_1(2l), \quad 0 \le l \le N/4-1 \\ x_4(l) = x_1(2l+1), \quad 0 \le l \le N/4-1 \\ X_3(k) = \sum_{l=0}^{N/4-1} x_3(l) W_{N/4}^{kl} = \mathrm{DFT}[x_3(l)] \\ X_4(k) = \sum_{l=0}^{N/4-1} x_4(l) W_{N/4}^{kl} = \mathrm{DFT}[x_4(l)] \end{cases}$$

$$\begin{cases} x_5(l) = x_2(2l), \quad 0 \le l \le N/4-1 \\ x_6(l) = x_2(2l+1), \quad 0 \le l \le N/4-1 \\ X_5(k) = \sum_{l=0}^{N/4-1} x_5(l) W_{N/4}^{kl} = \mathrm{DFT}[x_5(l)] \\ X_6(k) = \sum_{l=0}^{N/4-1} x_6(l) W_{N/4}^{kl} = \mathrm{DFT}[x_6(l)] \end{cases}$$

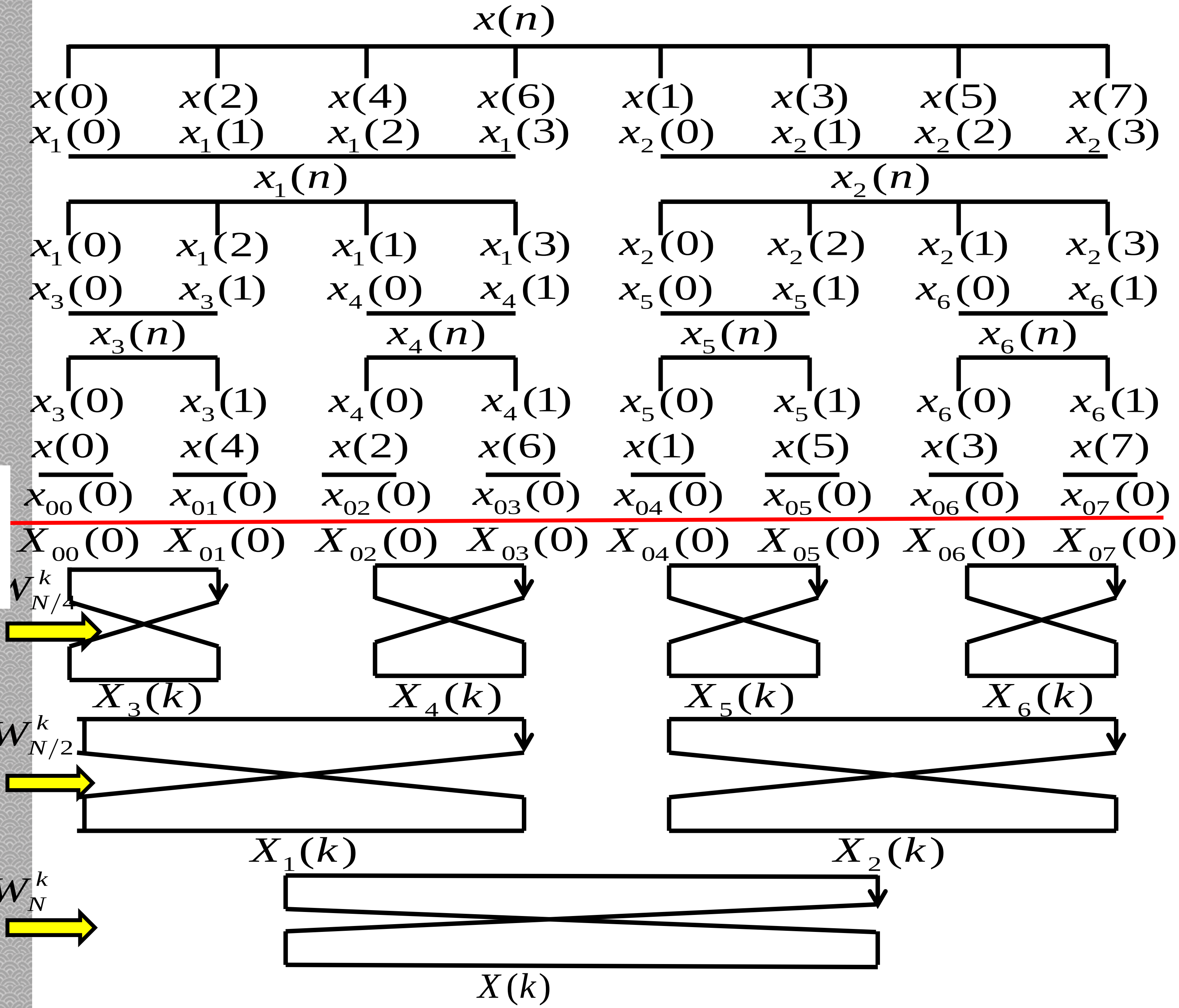


N点离散傅里叶变换的二次时域抽取运算流图(N=8)

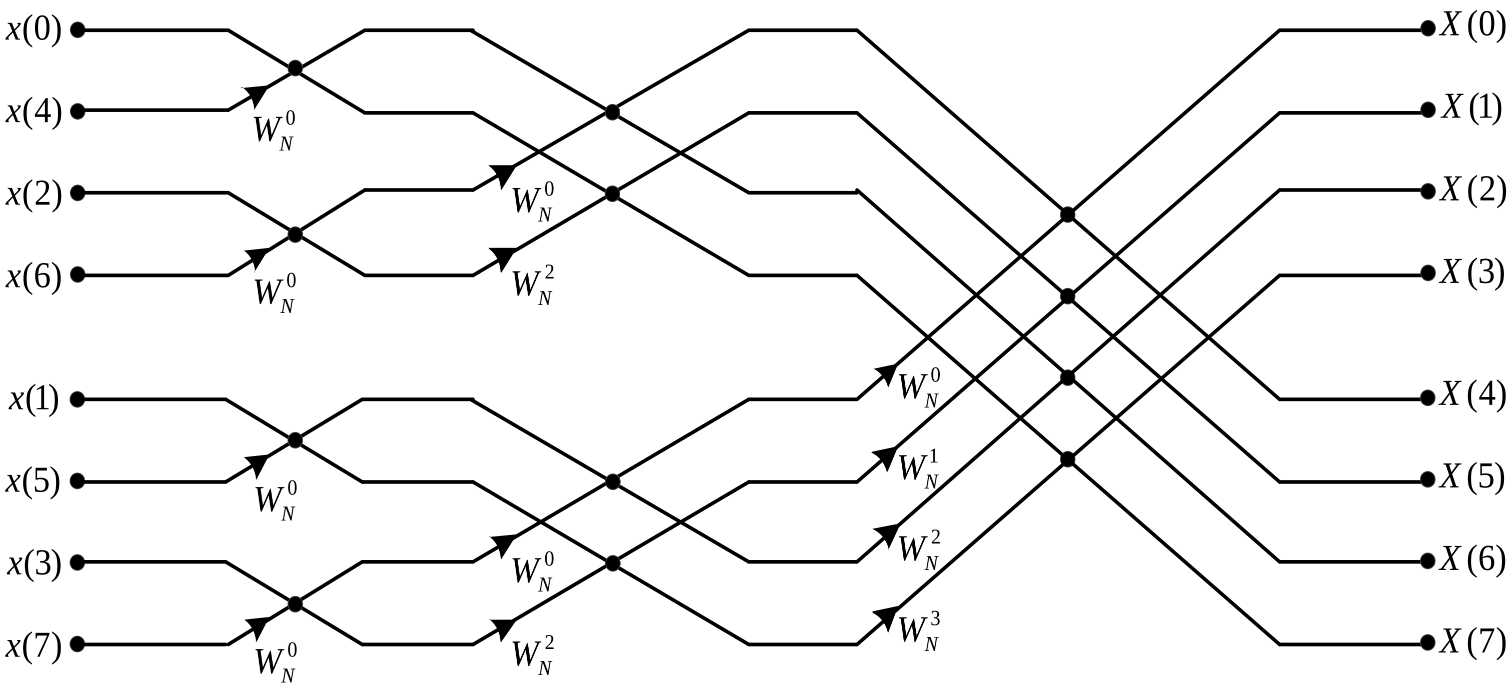


N点离散傅里叶变换的三次时域抽取运算流图(N=8)

N个1点DFT



$x(n)$
$x(0)$ $x(2)$ $x(4)$ $x(6)$ $x(1)$ $x(3)$ $x(5)$ $x(7)$
$x_1(0)$ $x_1(1)$ $x_1(2)$ $x_1(3)$ $x_2(0)$ $x_2(1)$ $x_2(2)$ $x_2(3)$
$x_1(n)$ $x_2(n)$
$x_1(0)$ $x_1(2)$ $x_1(1)$ $x_1(3)$ $x_2(0)$ $x_2(2)$ $x_2(1)$ $x_2(3)$
$x_3(0)$ $x_3(1)$ $x_4(0)$ $x_4(1)$ $x_5(0)$ $x_5(1)$ $x_6(0)$ $x_6(1)$
$x_3(n)$ $x_4(n)$ $x_5(n)$ $x_6(n)$
$x_3(0)$ $x_3(1)$ $x_4(0)$ $x_4(1)$ $x_5(0)$ $x_5(1)$ $x_6(0)$ $x_6(1)$
$x(0)$ $x(4)$ $x(2)$ $x(6)$ $x(1)$ $x(5)$ $x(3)$ $x(7)$
$x_{00}(0)$ $x_{01}(0)$ $x_{02}(0)$ $x_{03}(0)$ $x_{04}(0)$ $x_{05}(0)$ $x_{06}(0)$ $x_{07}(0)$
$X_{00}(0)$ $X_{01}(0)$ $X_{02}(0)$ $X_{03}(0)$ $X_{04}(0)$ $X_{05}(0)$ $X_{06}(0)$ $X_{07}(0)$
$W_{N/4}^{k}$
$X_3(k)$ $X_4(k)$ $X_5(k)$ $X_6(k)$
$W_{N/2}^{k}$
$X_1(k)$ $X_2(k)$
W_N^k
$X(k)$



N点DIT-FFT运算流图(N＝8)

分析： N点DIT-FFT 包含

M级蝶形运算，每一级包含N/2个碟形单元

一个蝶形单元需要 一次复数乘法，两次复数加法

$$C_M = M\frac{N}{2} = \frac{N}{2}\log_2 N \qquad C_A = M\frac{N}{2}\times 2 = N\log_2 N$$

DIT-FFT与直接计算DFT运算量比较

直接计算 需要N^2次复数乘法　$N(N-1)$次复数加法

DIT-FFT

$$\begin{cases} C_M = M\dfrac{N}{2} = \dfrac{N}{2}\log_2 N & \text{复数乘法} \\ C_A = M\dfrac{N}{2}\times 2 = N\log_2 N & \text{复数加法} \end{cases}$$

加速

$$R = \frac{N^2}{(N/2)\log_2 N} = \frac{2N}{\log_2 N}$$

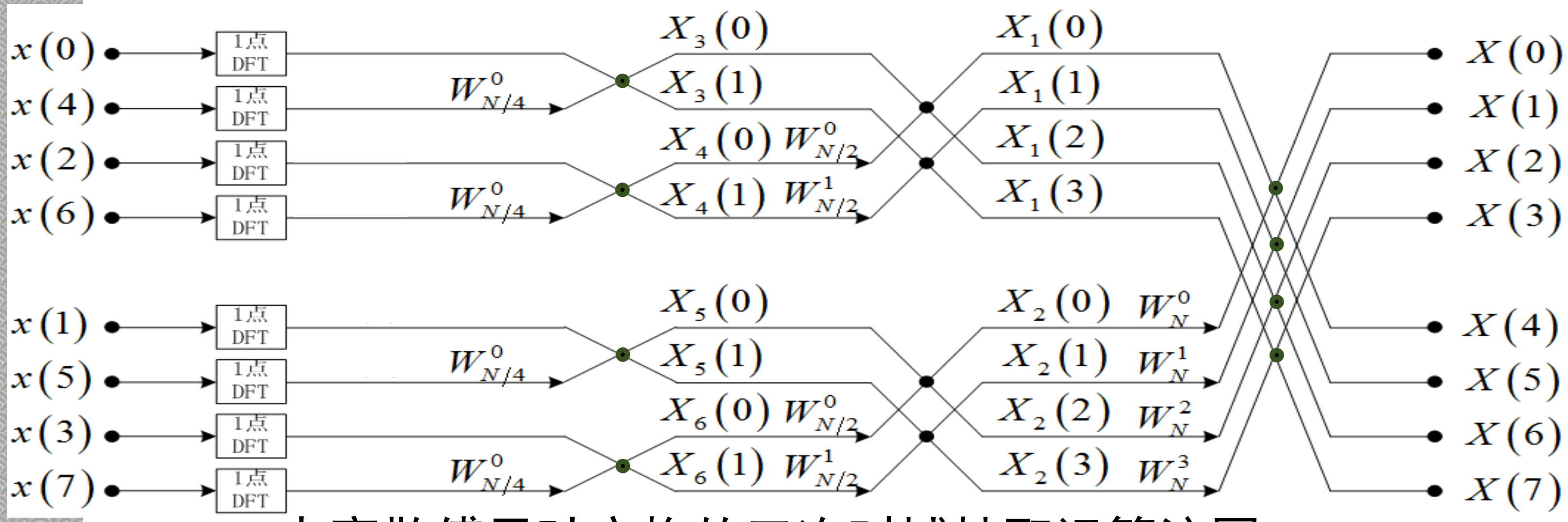
复数乘法加速倍数

MATLAB快速傅里叶变换调用格式： Xk=fft(xn,N)

5.1.1 时域抽取基2FFT算法

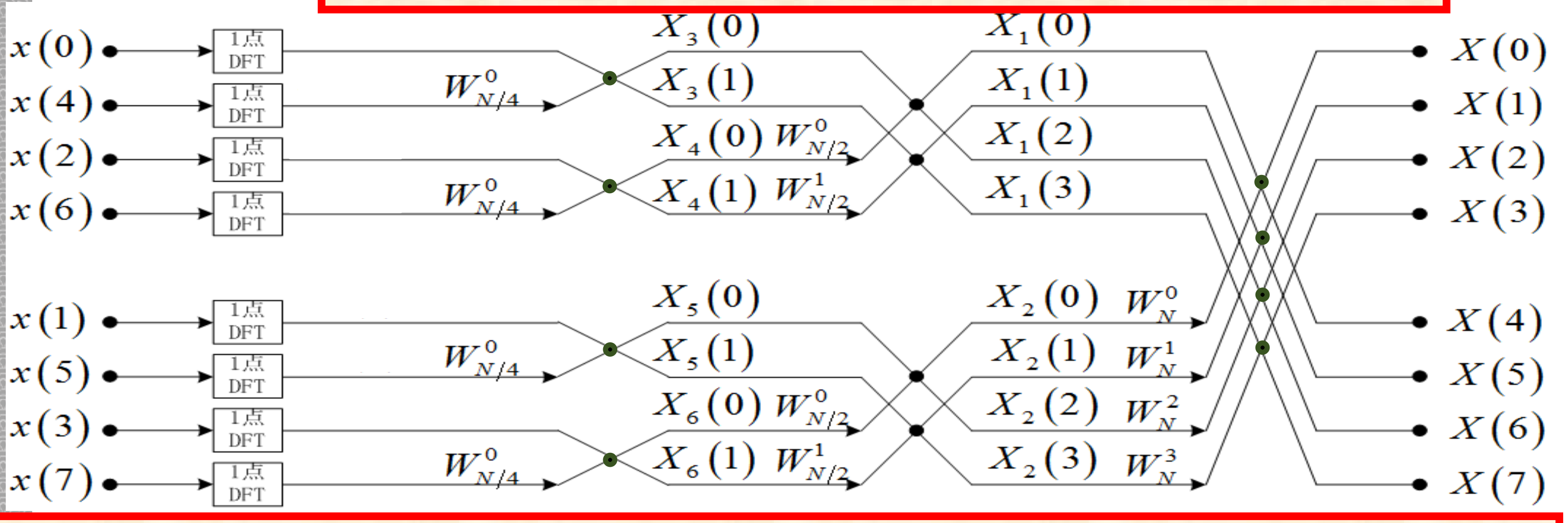
运算规律

1. 原位计算：运算过程无需增加新的存储单元



N点离散傅里叶变换的三次时域抽取运算流图(N=8)

运算规律 2. 旋转因子的变化规律和蝶形节点间距



对 $N=2^M$ 基2DIT-FFT算法，各级蝶形运算旋转因子和蝶形节点间距为

第一级蝶形运算(L=1)旋转因子为 W_N^0，蝶形节点间距为1

第二级蝶形运算(L=2)旋转因子为 W_N^0、$W_N^{N/4}$，蝶形节点间距为2

第三级蝶形运算(L=3)旋转因子为 W_N^0、$W_N^{N/8}$、$W_N^{2N/8}$、$W_N^{3N/8}$，蝶形节点间距为4

第M 级蝶形运算(L=M)旋转因子为 W_N^0、W_N^1、W_N^2，…$W_N^{(N/2-1)}$，

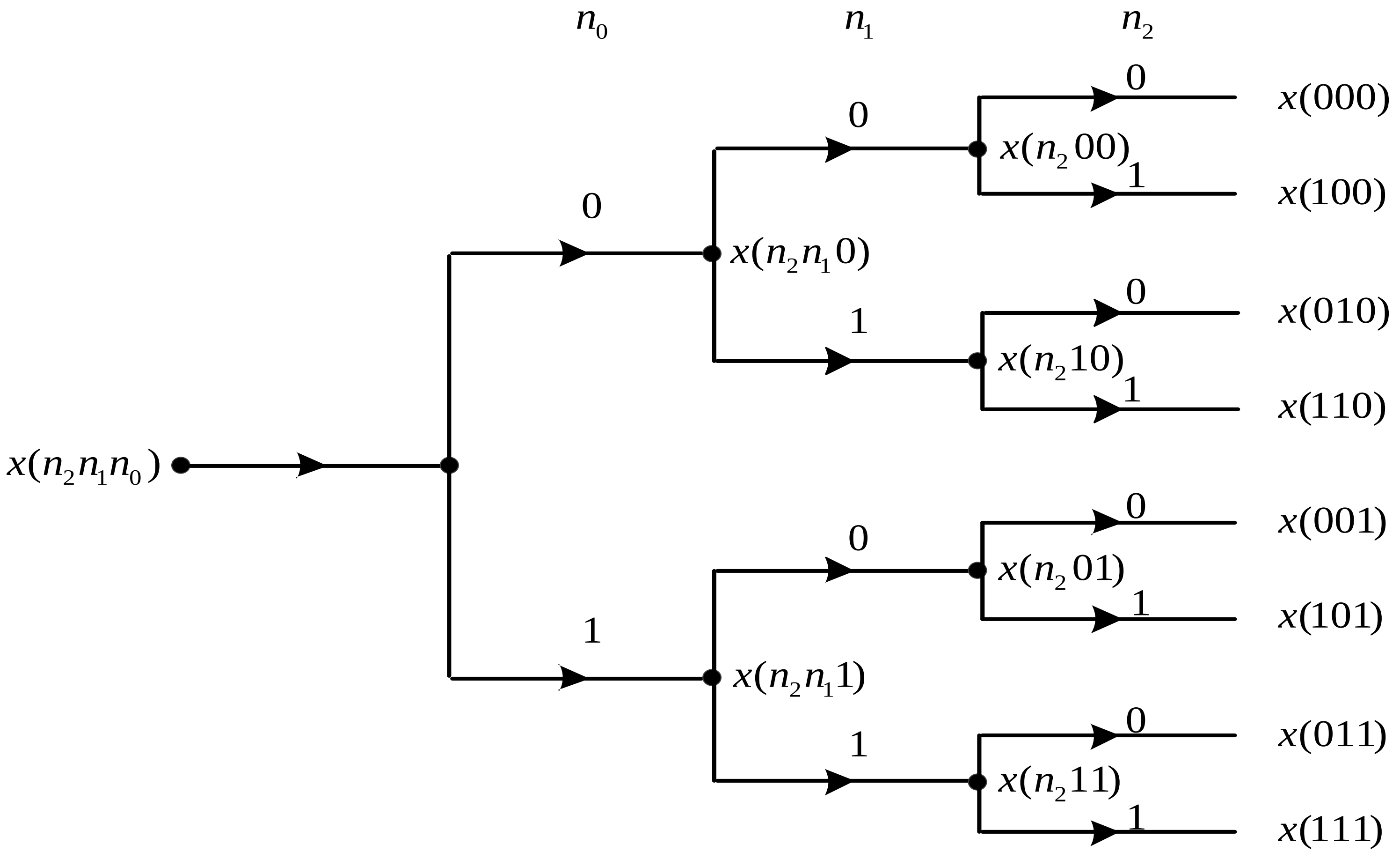
蝶形节点间距为 $N/2$

运算规律

3. 输入序列的倒序

	倒序	DFT		自然序
0	000	⟶	0	000
4	100	⟶	1	001
2	010	⟶	2	010
6	110	⟶	3	011
1	001	⟶	4	100
5	101	⟶	5	101
3	011	⟶	6	110
7	111	⟶	7	111

输入序列的倒序



n_0
n_1
n_2
$x(n_2n_1n_0)$
0
1
$x(n_2n_10)$
$x(n_2n_11)$
0
1
0
1
$x(n_200)$
$x(n_210)$
$x(n_201)$
$x(n_211)$
0
1
0
1
0
1
0
1
$x(000)$
$x(100)$
$x(010)$
$x(110)$
$x(001)$
$x(101)$
$x(011)$
$x(111)$

第五章 快速傅里叶变换(FFT)

- 5.1 基2FFT算法
  - 5.1.1 时域抽取基2FFT算法
  - 5.1.2 频域抽取基2FFT算法
- 5.2 IDFT的快速算法
- 5.3 实序列的FFT算法

5.1.2 频域抽取基2FFT算法

$x(n)\quad N=2^M\quad M$为正整数

按序列号n的自然排序将$x(n)$前后对半分

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

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

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

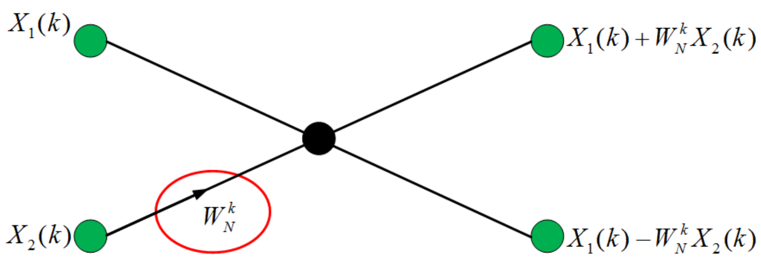
$$W_N^{kN/2}=e^{-j\frac{2\pi}{N}k\frac{N}{2}}=e^{-jk\pi}=\begin{cases}1 & k=even\\ -1 & k=odd\end{cases}$$

$$X(k)=\mathrm{DFT}\left[x(n)\right]=\begin{cases}X(2r)=\sum_{n=0}^{N/2-1}\left[\overbrace{x(n)+x(n+\frac{N}{2})}^{x_1(n)}\right]\bullet W_N^{2rn} & k=even\\ X(2r+1)=\sum_{n=0}^{N/2-1}\left[\underbrace{x(n)-x(n+\frac{N}{2})}_{x_2(n)}\right]W_N^{n}\bullet W_N^{2rn} & k=odd\end{cases}$$

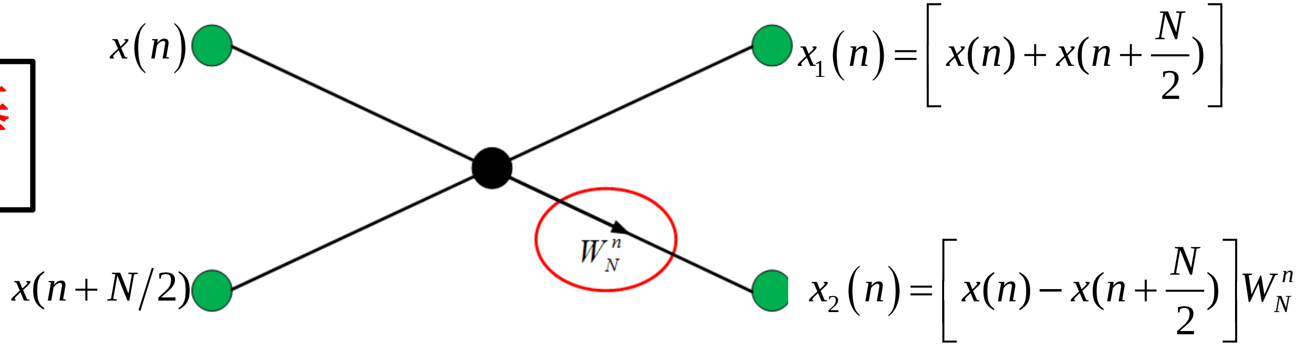
$$X(k)=\mathrm{DFT}\left[x(n)\right]=\begin{cases}\mathrm{DFT}\left[x_1(n)\right]\\ \mathrm{DFT}\left[x_2(n)\right]\end{cases}=\begin{cases}X(2r)=\sum_{n=0}^{N/2-1}x_1(n)\bullet W_N^{2rn} & k=even \quad X_1(k)\\ X(2r+1)=\sum_{n=0}^{N/2-1}x_2(n)\bullet W_N^{2rn} & k=odd \quad X_2(k)\end{cases}$$

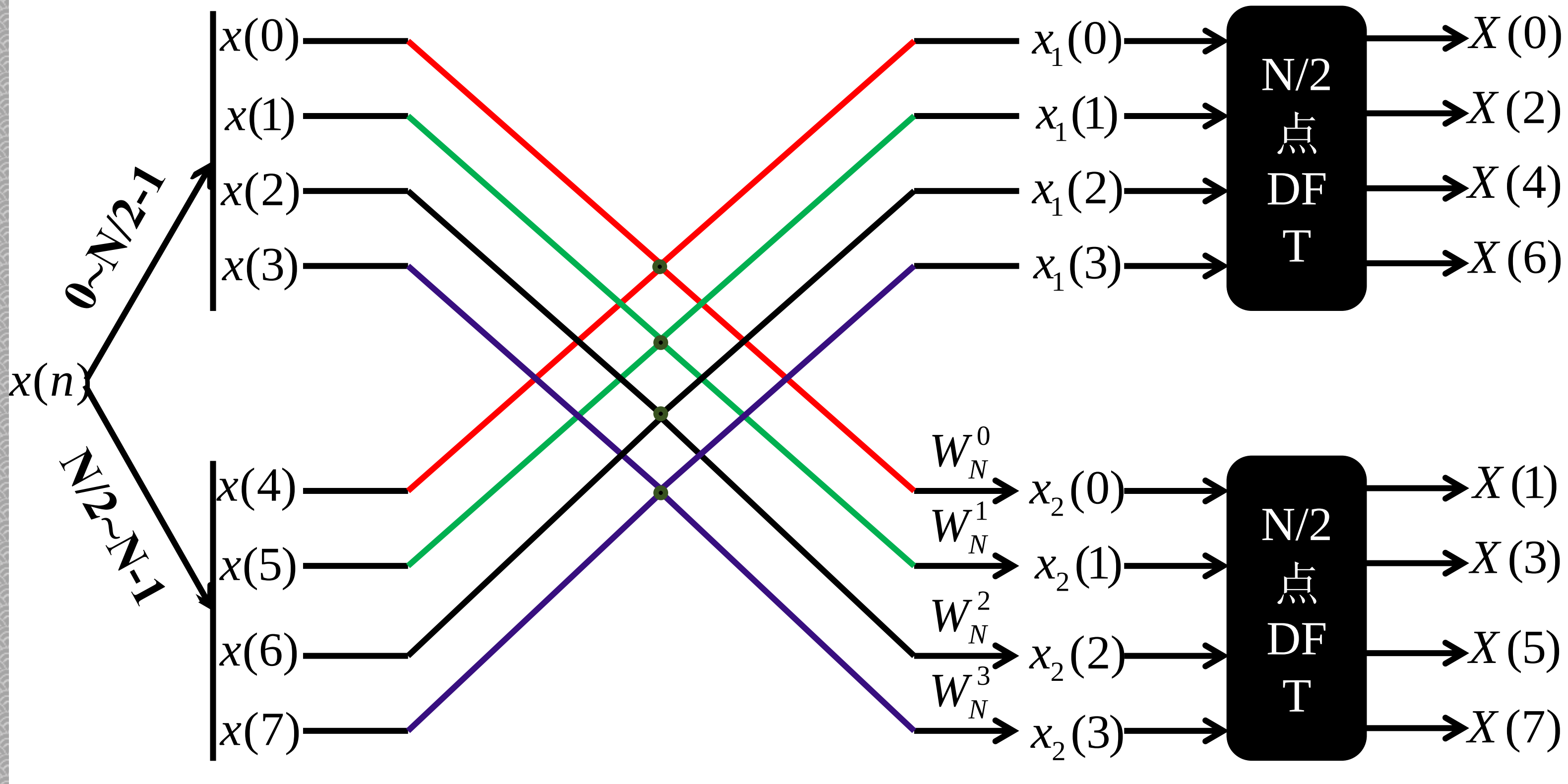
$$x_1(n)=\left[x(n)+x(n+\frac{N}{2})\right] \qquad x_2(n)=\left[x(n)-x(n+\frac{N}{2})\right]W_N^n$$

时域抽取基2FFT算法



频域抽取基2FFT算法

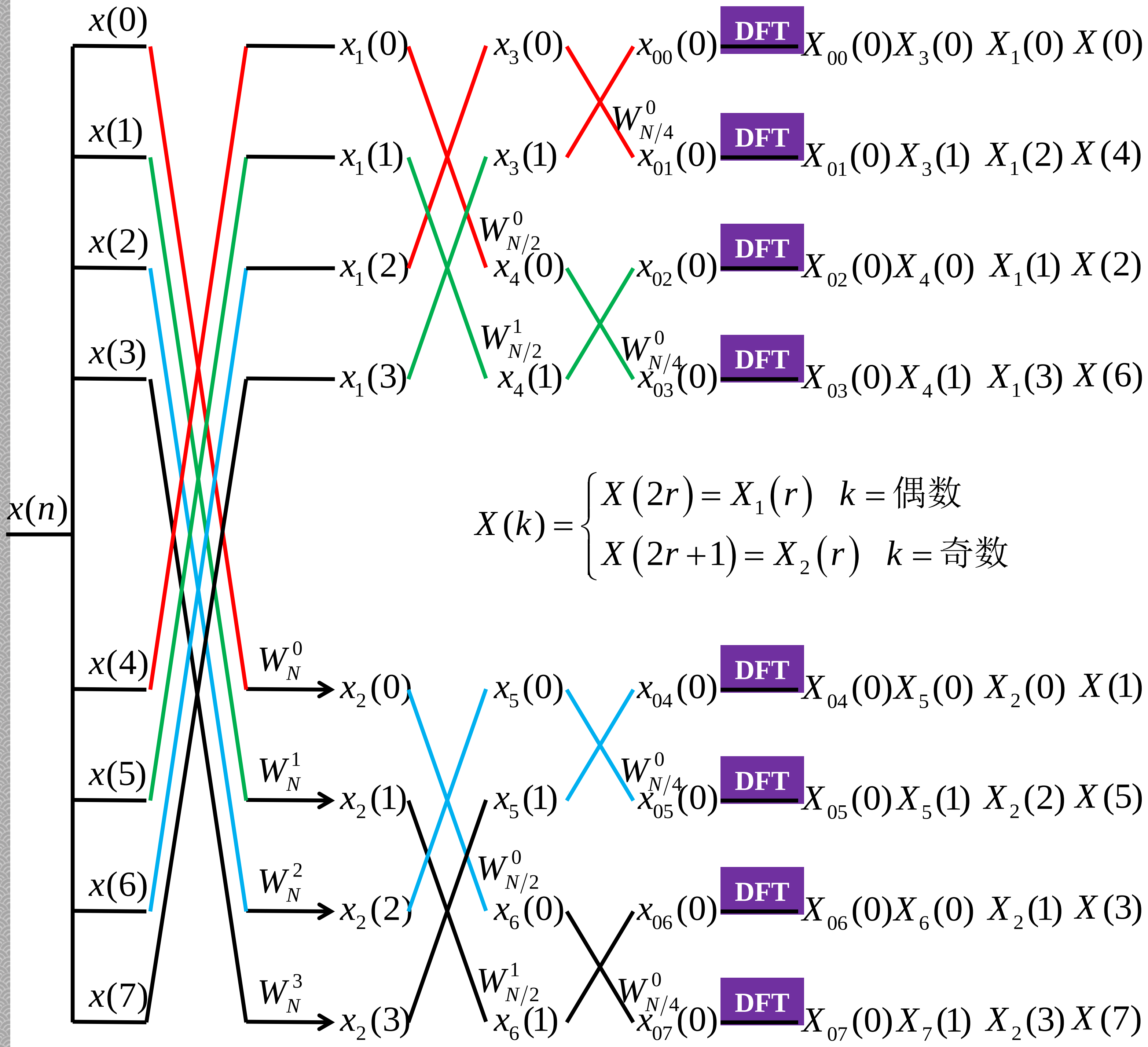




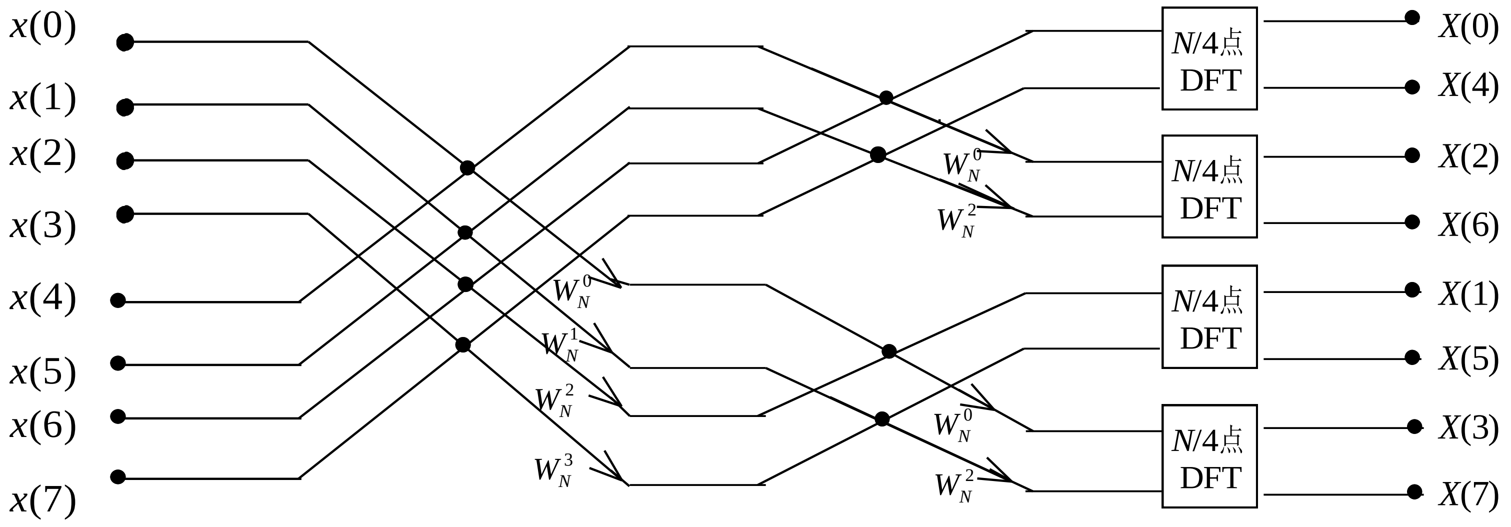
$$X(2r) = \sum_{n=0}^{N/2-1} x_1(n) \cdot W_N^{2rn} = \sum_{n=0}^{N/2-1} x_1(n) \cdot W_{N/2}^{rn}$$ $X_1(n)$的$N/2$点DFT

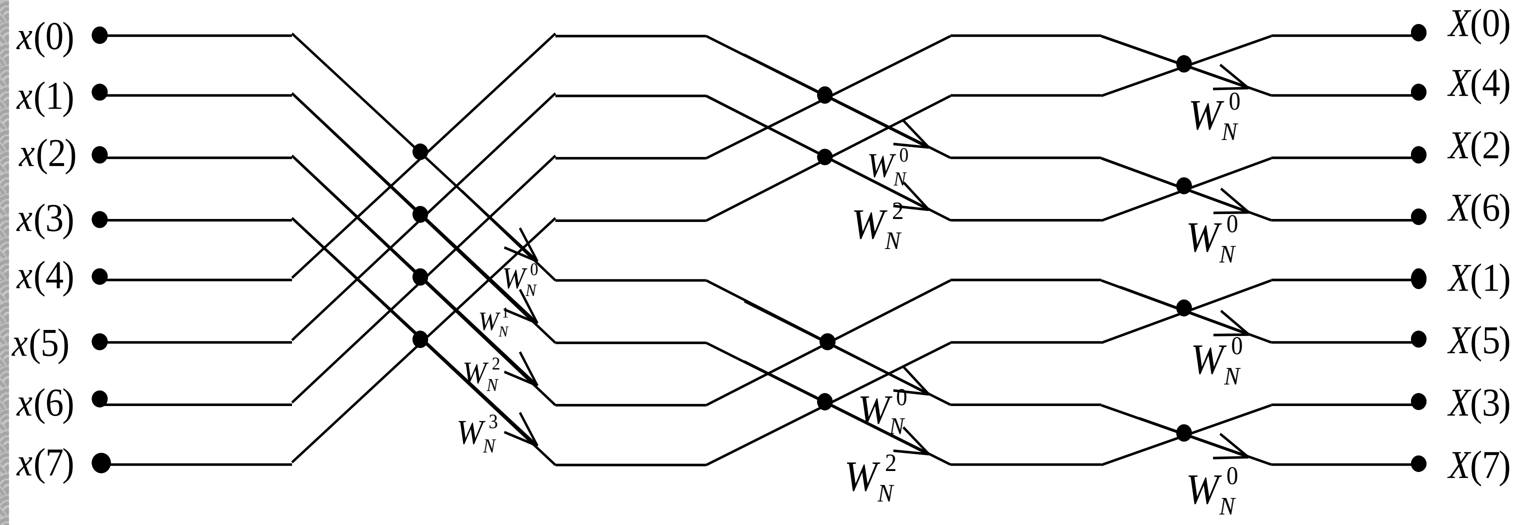
$$X(2r+1) = \sum_{n=0}^{N/2-1} x_2(n) \cdot W_N^{2rn} = \sum_{n=0}^{N/2-1} x_2(n) \cdot W_{N/2}^{rn}$$ $X_2(n)$的$N/2$点DFT

$r = 0, 1, \ldots, N/2-1$



$$X(k)=\begin{cases} X(2r)=X_1(r) & k=\text{偶数} \\ X(2r+1)=X_2(r) & k=\text{奇数} \end{cases}$$





特点分析： N个1点DFT M级蝶形运算 N/2个碟算 每个蝶算2加1乘

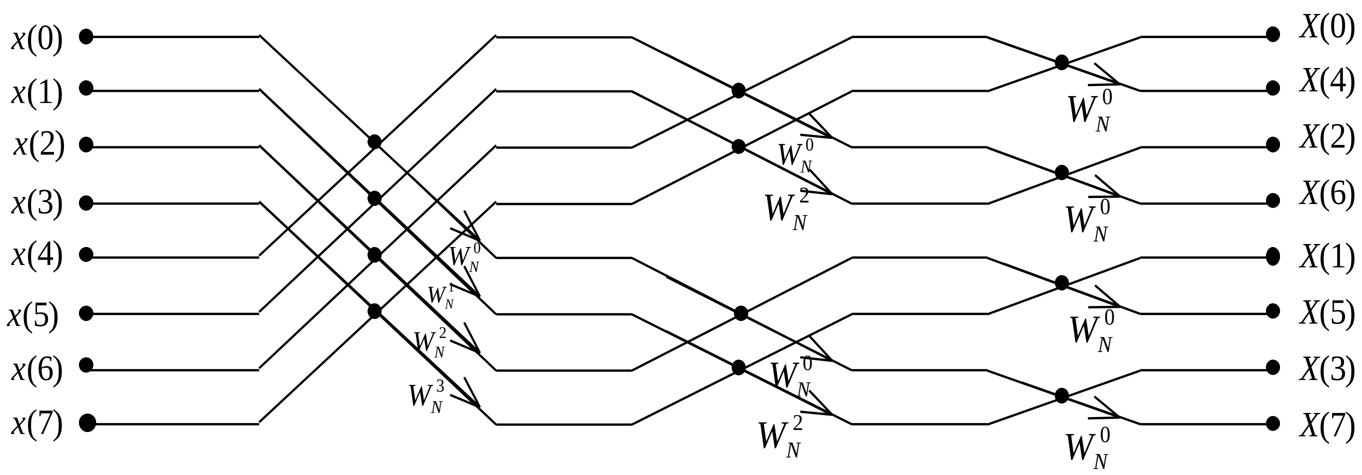
运算量： $C_M = M\dfrac{N}{2} = \dfrac{N}{2}\log_2 N \quad C_A = M\dfrac{N}{2}\times 2 = N\log_2 N$

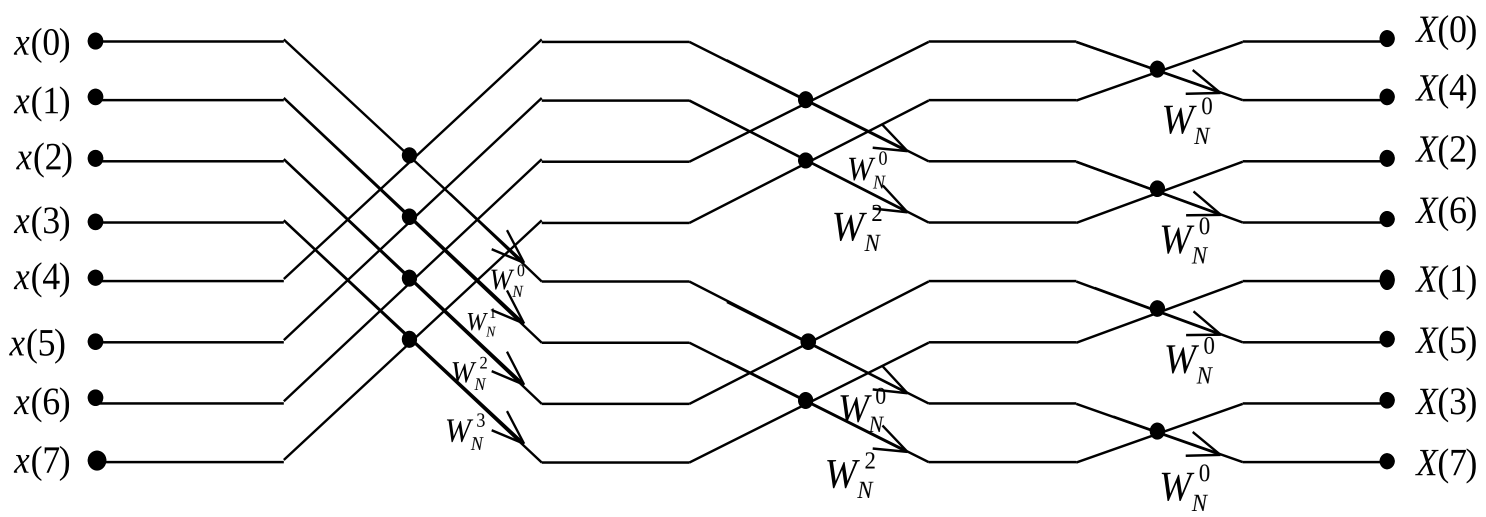
运算规律

原位计算：运算过程无需增加新的存储单元

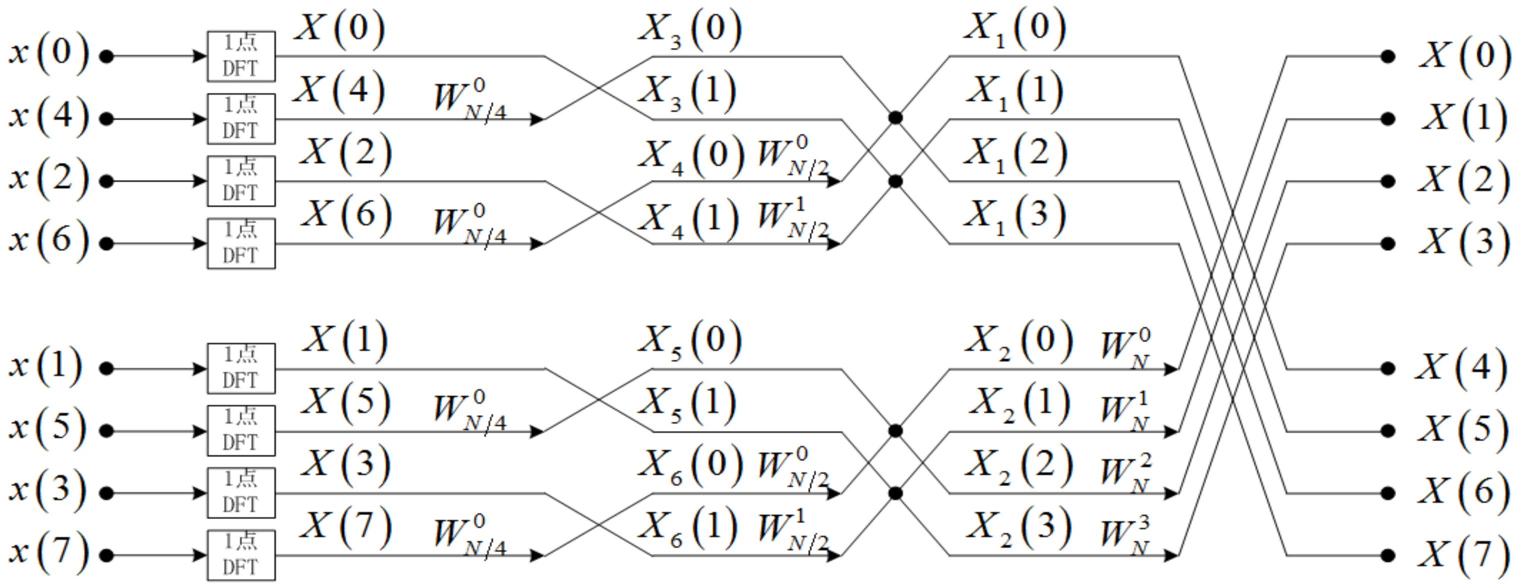
输入序列的倒序

旋转因子的变化规律和蝶形节点间距





x(0)
x(1)
x(2)
x(3)
x(4)
x(5)
x(6)
x(7)
W_N^0
W_N^1
W_N^2
W_N^3
W_N^0
W_N^2
W_N^0
W_N^2
W_N^0
W_N^0
W_N^0
W_N^0
X(0)
X(4)
X(2)
X(6)
X(1)
X(5)
X(3)
X(7)



x(0)
x(4)
x(2)
x(6)
x(1)
x(5)
x(3)
x(7)
1点 DFT
X(0)
X(4)
X(2)
X(6)
X(1)
X(5)
X(3)
X(7)
$W_{N/4}^0$
$X_3(0)$
$X_3(1)$
$X_4(0)$
$X_4(1)$
$X_5(0)$
$X_5(1)$
$X_6(0)$
$X_6(1)$
$W_{N/2}^0$
$W_{N/2}^1$
$X_1(0)$
$X_1(1)$
$X_1(2)$
$X_1(3)$
$X_2(0)$
$X_2(1)$
$X_2(2)$
$X_2(3)$
W_N^0
W_N^1
W_N^2
W_N^3
X(0)
X(1)
X(2)
X(3)
X(4)
X(5)
X(6)
X(7)

➢仔细观察基2DIF-FFT运算流图和基2DIT-FFT运算流图会发现，将频域抽取法的运算流图反转，并将输入变输出，输出变输入，正好得到时域抽取法的运算流图

➢按频域抽取算法与按时域抽取算法是两种等价的FFT算法，此外，在基2FFT的基础上，还有变形基2FFT运算流图，原理类似

第五章 快速傅里叶变换(FFT)

- 5.1 基2FFT算法
  - 5.1.1 时域抽取基2FFT算法
  - 5.1.2 频域抽取基2FFT算法
- 5.2 IDFT的快速算法
- 5.3 实序列的FFT算法

5.2 IDFT的快速算法

$$DFT / IDFT \Rightarrow \begin{cases} X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} & 0 \le k \le N-1 \\ x(n) = \dfrac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn} & 0 \le n \le N-1 \end{cases}$$

旋转因子指数变极性法

直接调用FFT子程序法1

直接调用FFT子程序法2

5.2 IDFT的快速算法

上述FFT算法流图也可以用于离散傅里叶逆变换(Inverse Discrete Fourier Transform，简称IDFT)。比较DFT和IDFT运算公式：

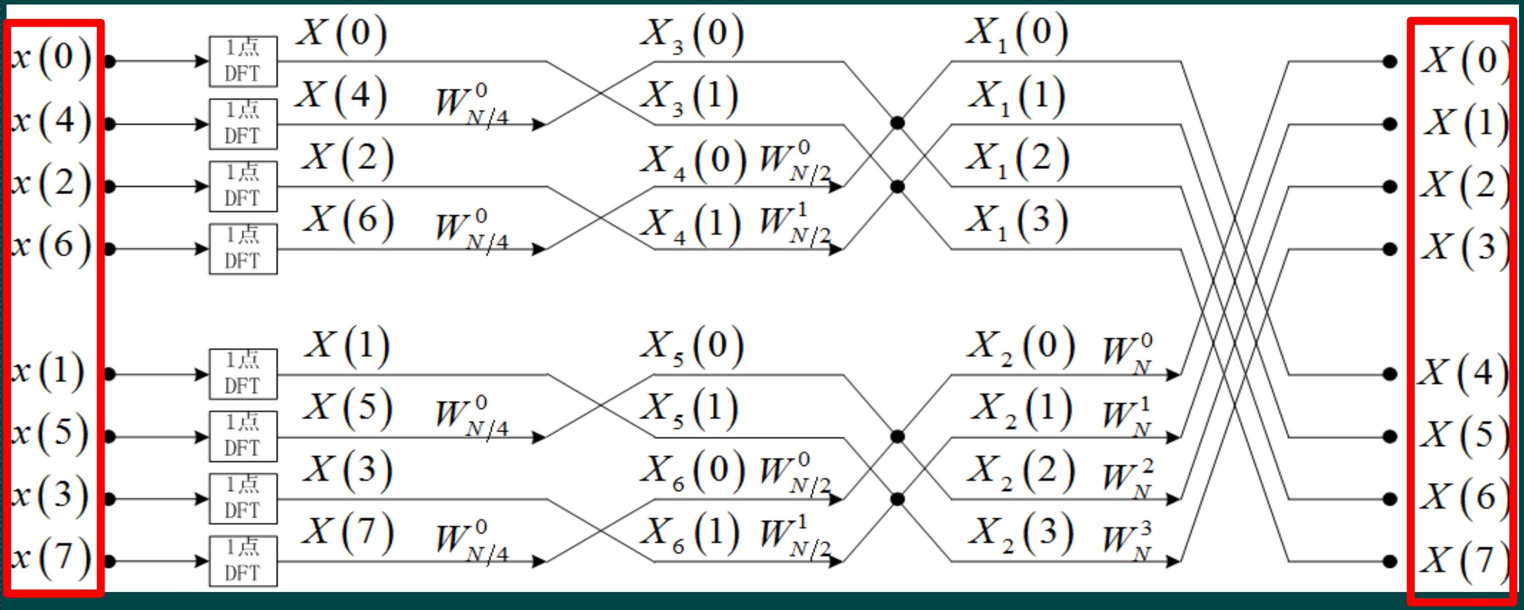
$$X(k) = DFT[x(n)] = \sum_{n=0}^{N-1} x(n) W_N^{kn}$$

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

比较上面两式, 只要将DFT运算式中的旋转因子W_N^{kn}变为W_N^{-kn},即旋转因子指数变极性, $X(k)$作为输入序列,并乘以系数1/N, 就成为IDFT的运算公式

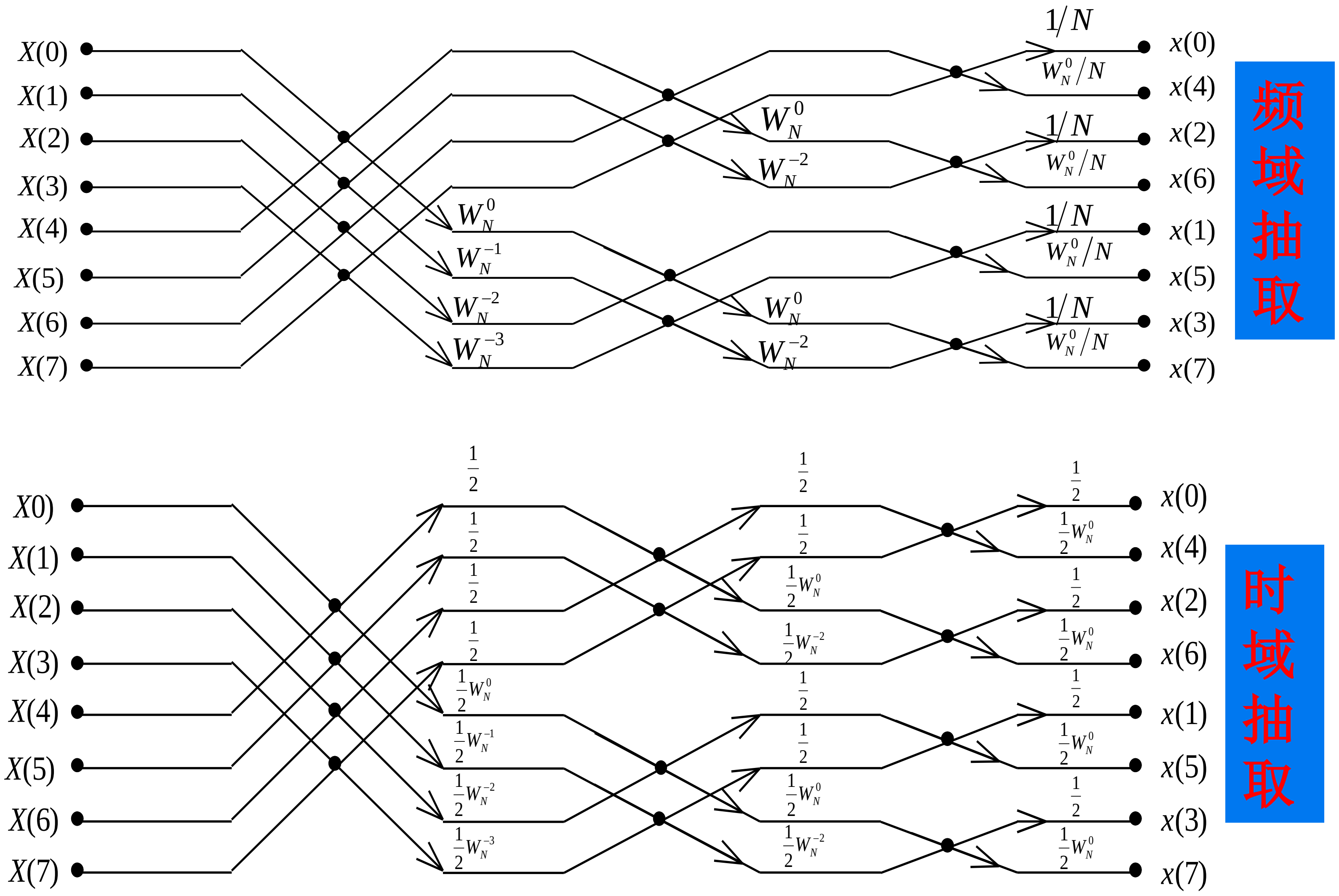
$$X(k) = DFT[x(n)] = \sum_{n=0}^{N-1} x(n) W_N^{kn} \qquad x(n) = IDFT[x(n)] = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn}$$

所以，如果将基2DIT-FFT或基2DIF-FFT算法中的旋转因子 W_N^p 改为 W_N^{-p}，把$X(k)$作为输入序列，运算结果乘以$1/N$，就可以用来快速计算IDFT，得到输出序列 $x(n)$



旋转因子指数变极性法

$$W_N^{kn} \Rightarrow W_N^{-kn}$$



直接调用FFT子程序法

➢直接调用FFT子程序求IFFT,就是将$X(k)$或者$X(k)$的变体作为FFT程序的输入，所以我们需要求出$x(n)$与DFT[$X(k)$]的关系

直接调用FFT子程序 方法1

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

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

上式两边取共轭得

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

可用 *fft* 子程序实现DFT

符合$X^*(k)$的DFT的定义式

因此，将$X(k)$取共轭后得$X^*(k)$，输入FFT算出DFT[$X^*(k)$]，对其取共轭，再乘以$1/N$得$x(n)$

直接调用FFT子程序 方法2

$X(k)$的DFT $\quad g(n)=\sum_{k=0}^{N-1} X(k)W_N^{kn}=DFT[X(k)]$

$$0\le n\le N-1$$

根据DFT的时域位移性质得

$$\frac{1}{N}g(N-n)=\frac{1}{N}\sum_{k=0}^{N-1}X(k)W_N^{k(N-n)}=\frac{1}{N}\sum_{k=0}^{N-1}X(k)W_N^{kN}W_N^{-kn}$$

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

$x(n)$的IDFT的定义式

$$x(n)=\frac{1}{N}g(N-n) \qquad 0\le n\le N-1$$

将$X(k)$作为FFT的输入，求出$g(n)$，将$g(n)$进行移位得$g(N\text{-}n)$，再乘以$1/N$，得$x(n)$，即求出$X(k)$的逆变换

第五章 快速傅里叶变换(FFT)

- 5.1 基2FFT算法
  - 5.1.1 时域抽取基2FFT算法
  - 5.1.2 频域抽取基2FFT算法
  - 5.1.3 其他快速算法简介
- 5.2 IDFT的快速算法
- 5.3 实序列的FFT算法

5.3实序列的FFT算法

三种情况：

- 把实序列$x(n)$看作虚部为零的复序列，直接调用fft；
- 对两个N点的实序列$x(n)$和$h(n)$，构成复序列$y(n)$， 即

$$y(n)=x(n)+jh(n), n=0,1,\cdots,N-1$$

时间浪费

- 对$y(n)$进行N点FFT，输出$Y(k)$，则

$$\left.\begin{aligned}X(k)&=DFT[x(n)]=Y_{ep}(k)=\frac{1}{2}[Y(k)+Y^*(N-k)]\\ H(k)&=DFT[h(n)]=-jY_{op}(k)=\frac{1}{2j}[Y(k)-Y^*(N-k)]\end{aligned}\right\},k=0,1,\cdots,N-1$$

运算时间略多于一个N点FFT，可获得两个N点实序列的FFT

- 设$x(n)$为N点实序列，取$x(n)$的偶数点和奇数点分别作为新构造序列$y(n)$的实部和虚部，即

$$x_1(n) = x(2n), x_2(n) = x(2n+1), n = 0,1,\cdots, N/2-1$$
$$y(n) = x_1(n) + jx_2(n), n = 0,1,\cdots, N/2-1$$

对$y(n)$进行$N/2$点FFT，输出$Y(k)$，则

$$\left.\begin{aligned} X_1(k) &= DFT[x_1(n)] = Y_{ep}(k) \\ X_2(k) &= DFT[x_2(n)] = -jY_{op}(k) \end{aligned}\right\}, k = 0,1,\cdots,\frac{N}{2}-1$$

根据DIT-FFT的思想及下式，可得到

$$X(k) = X_1(k) + W_N^k X_2(k), \quad k = 0,1,\cdots, N/2-1$$

由于$x(n)$为实序列，所以$X(k)$具有共轭对称性，$X(k)$另外$N/2$点的值为

$$X(N-k)=X^*(k), k=1,2,\cdots,N/2-1$$

运算速度提高近一倍

运算效率（复乘）：原蝶形/现在

$$\eta=\frac{M\frac{N}{2}}{(M-1)\frac{N}{4}+\frac{N}{2}}=\frac{2M}{M+1}\approx 2$$