第四讲 FFT
快速傅里叶变换

目录

§4-1引言

一. DFT的计算工作量

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

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

两者的差别仅在指数的符号和因子1/N.

一个X(k)的值的工作量，如X(1)

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

通常x(n)和 W_N^{nk}都是复数,所以计算一个X(k)的值需要N次复数乘法运算,和 次$N-1$复数加法运算.

那么,所有的X(k)就要N^2次复数乘法运算,N(N-1)次复数加法运算.当N很大时,运算量将是惊人的,如N=1024,则要完成1048576 次(一百多万次)复数乘法运算.

这样,难以做到实时处理.

二.改进的途径

1. W_N^{nk} 的对称性和周期性

对称性： $(W_N^{nk})^* = W_N^{-nk}$，

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

得： $W_N^{n(N-k)} = W_N^{(N-n)k} = W_N^{-nk} \quad (\because W_N^{Nk} = W_N^{Nn} = e^{-2\pi k(n)} = 1)$,

$W_N^{N/2} = -1 \quad (\because W_N^{N/2} = e^{-j\pi} = -1)$,

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

利用上述特性，可以将有些项合并，并将DFT分解为短序列，从而降低运算次数，提高运算速度. 1965年，库利(cooley)和图基(Tukey)首先提出FFT算法. 对于N点DFT，仅需$(N/2)\log_2 N$ 次复数乘法运算. 例如$N=1024=2^{10}$ 时，需要$(1024/2)\log_2 2^{10} =512*10=5120$次。

5120/1048576=4.88% ，速度提高20倍

§4-2 按时间抽取(DIT)的FFT算法
—库利-图基算法

一. 算法原理(基2FFT)

(一)N/2点DFT

1. 先将$x(n)$按n的奇偶分为两组作DFT, 设N=2^L , 不足时, 可补些零。

这样有:

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

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

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

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

(n为偶数)　　(n为奇数)

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

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

由于：$$W_N^2=e^{-j\frac{2\pi}{N}*2}=e^{-j2\pi/(\frac{N}{2})}=W_{\frac{N}{2}}$$

所以，上式可表示为：

$$X(k)=\sum_{r=0}^{\frac{N}{2}-1}x_1(r)W_{\frac{N}{2}}^{rk}+W_N^k\sum_{r=0}^{\frac{N}{2}-1}x_2(r)W_{\frac{N}{2}}^{rk}=X_1(k)+W_N^kX_2(k)$$

其中，

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

2. 两点结论：

(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 $k=0, 1, \cdots, \frac{N}{2}-1$ 个；

即前一半的结果。

3. X(k)的后一半的确定

由于 $W_{\frac{N}{2}}^{r(k+\frac{N}{2})} = W_{\frac{N}{2}}^{rk}$ （周期性），所以：

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

同理，$X_2(\frac{N}{2}+k) = X_2(k)$

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 $W_N^{(\frac{N}{2}+k)} = W_N^{\frac{N}{2}} W_N^k = -W_N^k$ ，所以

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

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

可见，X(k)的后一半，也完全由$X_1(k)$，X_2 (k)的前一半所确定。

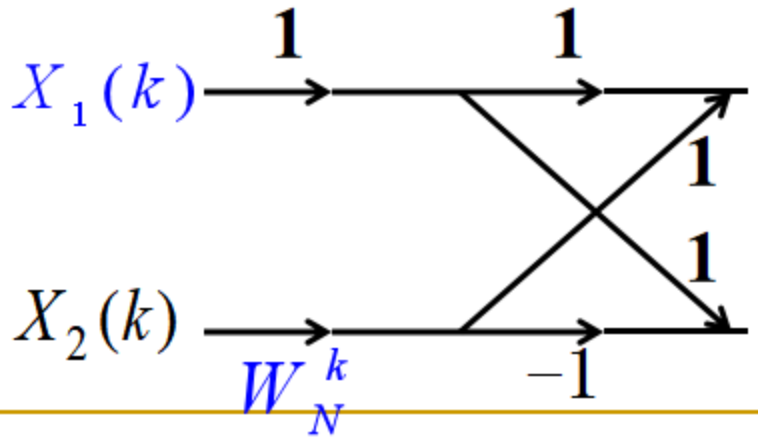
*N点的DFT可由两个N/2点的DFT来计算。

4. 蝶形运算

由$X_1(k)$、$X_2(k)$表示$X(k)$的运算是一种特殊的运算-碟形运算

$$\begin{cases} X(k)=X_1(k)+W_N^k X_2(k) & \text{前一半} \quad (k=0,1,\cdots,\frac{N}{2}-1) \\ X(k+\frac{N}{2})=X_1(k)-W_N^k X_2(k) & \text{后一半} \quad (k+\frac{N}{2}=\frac{N}{2},\cdots,N-1) \end{cases}$$

实现上式运算的流图称作蝶形运算 （N/2个蝶形）



$X(k)=X_1(k)+W_N^k X_2(k)$ （前一半）

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

5. 计算工作量分析

按奇、偶分组后的计算量：

(1) N/2点的DFT运算量：复乘次数：$(\frac{N}{2})^2 = \frac{N^2}{4}$

复加次数：$\frac{N}{2}(\frac{N}{2}-1)$

(2) 两个N/2点的DFT运算量：复乘次数：$\frac{N^2}{2}$

复加次数：$N(\frac{N}{2}-1)$

(3) N/2个蝶形运算的运算量：复乘次数：$\frac{N}{2}$

复加次数：$2\cdot\frac{N}{2} = N$

总共运算量：

复乘：$\frac{N^2}{2} + \frac{N}{2} \approx \frac{N^2}{2}$

复加：$N(\frac{N}{2}-1) + N = \frac{N^2}{2}$

*但是，N点DFT的复乘为N^2；复加N(N-1)；与分解后相比可知，计算工作点差不多减少 一半。

例如 N=8 时的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, 即 $x(0), x(2), x(4), x(6)$;

分别记作: $x_1(0) = x(0)$,

$x_1(1) = x(2)$,

$x_1(2) = x(4)$,

$x_1(3) = x(6)$;

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，分别记作：

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

进行$N/2=4$点的DFT，得$X_2(k)$

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

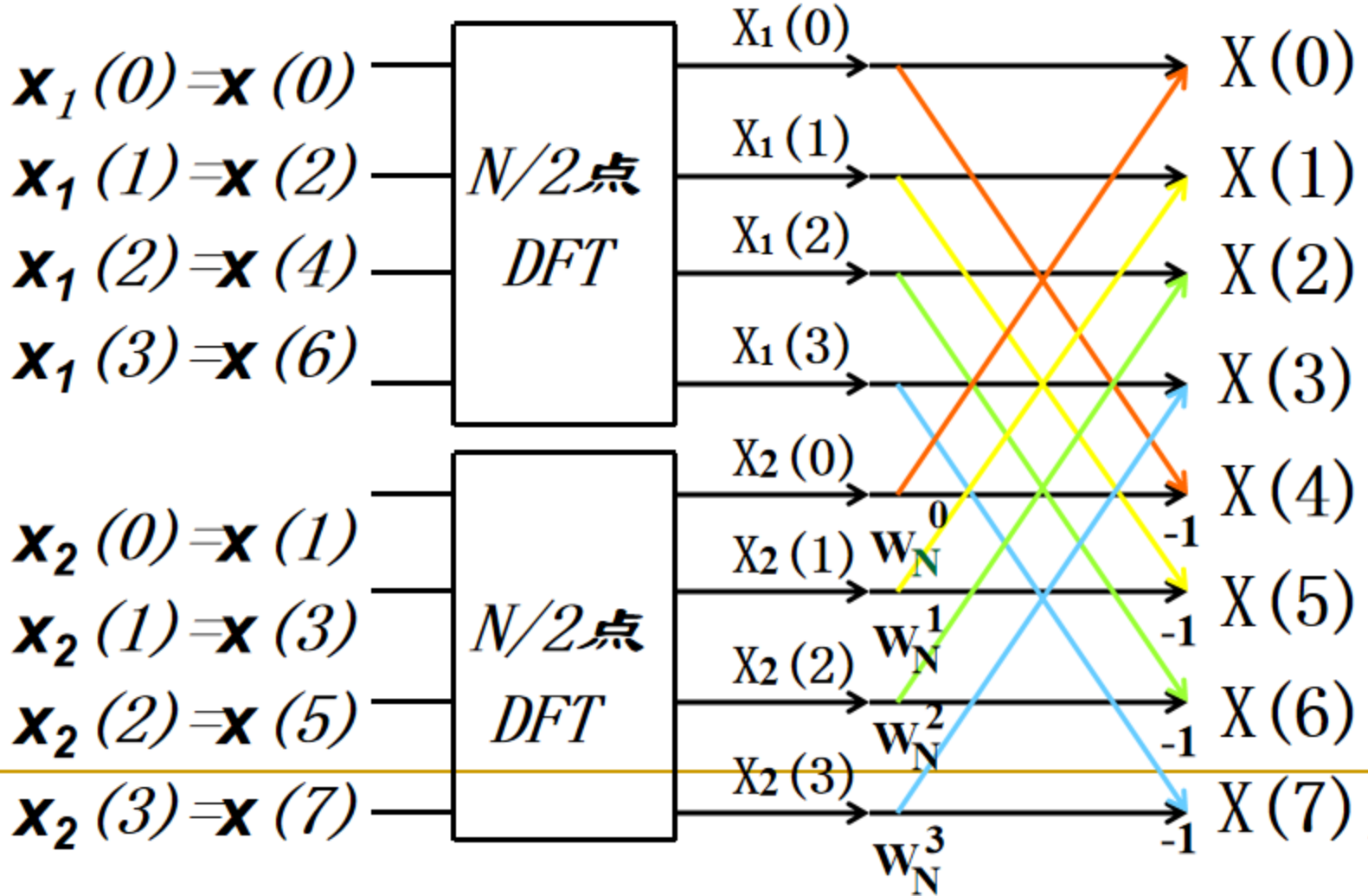
$$k=0,1,2,3$$

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

$$X(k+4)=X_1(k)-W_N^kX_2(k),k=0,1,2,3$$

(3) 对 $X_1(k)$ 和 $X_2(k)$ 进行蝶形运算，前半部为 $X(0)\sim X(3)$，后半部分为 $X(4)\sim X(7)$

整个过程如下图所示：



(二) N/4点DFT

由于N=2ᴸ，所以 N/2仍为偶数，可以进一步把每个N/2点的序列再按其奇偶部分分解为两个N/4的子序列。例如，n为偶数时的 N/2点分解为：

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

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

进行N/4点的DFT，得到

$$\begin{cases} X_3(k) = \sum_{l=0}^{\frac{N}{4}-1} x_3(l) W_{N/4}^{lk} = \sum_{l=0}^{\frac{N}{4}-1} x_1(2l) W_{N/2}^{2lk} & \text{(偶中偶)} \\ X_4(k) = \sum_{l=0}^{\frac{N}{4}-1} x_4(l) W_{N/4}^{lk} = \sum_{l=0}^{\frac{N}{4}-1} x_1(2l+1) W_{N/2}^{(2l+1)k} & \text{(偶中奇)} \end{cases}$$

从而可得到前N/4的$X_1(k)$

$$X_1(k) = X_3(k) + W_{\frac{N}{2}}^{k} X_4(k)$$

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

后N/4的$X_1(k)$为

$$X_1(\frac{N}{4} + k) = X_3(k) - W_{\frac{N}{2}}^{k} X_4(k)$$

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

同样对n为奇数时，$N/2$点分为两个$N/4$点的序列进行DFT，则有：

$$\begin{cases} X_5(k)=\sum_{l=0}^{\frac{N}{4}-1}x_2(2l)W_{N/4}^{lk}=\sum_{l=0}^{\frac{N}{4}-1}x_5(l)W_{N/4}^{lk} & \text{(奇中偶)} \\ X_6(k)=\sum_{l=0}^{\frac{N}{4}-1}x_2(2l+1)W_{N/4}^{lk}=\sum_{l=0}^{\frac{N}{4}-1}x_6(l)W_{N/4}^{lk} & \text{(奇中奇)} \end{cases}$$

由$X_5(k)$、$X_6(k)$进行碟形运算，得：

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

例如，N=8时的DFT可分解为四个N/4的DFT，

具体步骤如下：

(1) 将原序列$x(n)$的“偶中偶”部分：

$$x_3(l) = x_1(r) = x(n)$$

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

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

构成N/4点DFT，从而得到$X_3(0)$, $X_3(1)$。

(2) 将原序列$x(n)$的“偶中奇”部分:

$$x_4(l) = x_1(r) = x(n)$$

$$x_4(0) = x_1(1) = x(2)$$

$$x_4(1) = x_1(3) = x(6)$$

构成N/4点DFT，从而得到$X_4(0), X_4(1)$。

(3) 将原序列$x(n)$的“奇中偶”部分:

$$x_5(l) = x_2(r) = x(n)$$

$$x_5(0) = x_2(0) = x(1)$$

$$x_5(1) = x_2(2) = x(5)$$

构成N/4点DFT，从而得到$X_5(0), X_5(1)$。

(4) 将原序列$x(n)$的“奇中奇”部分:

$$x_6(l) = x_2(r) = x(n)$$

$$x_6(0) = x_2(1) = x(3)$$

$$x_6(1) = x_2(3) = x(7)$$

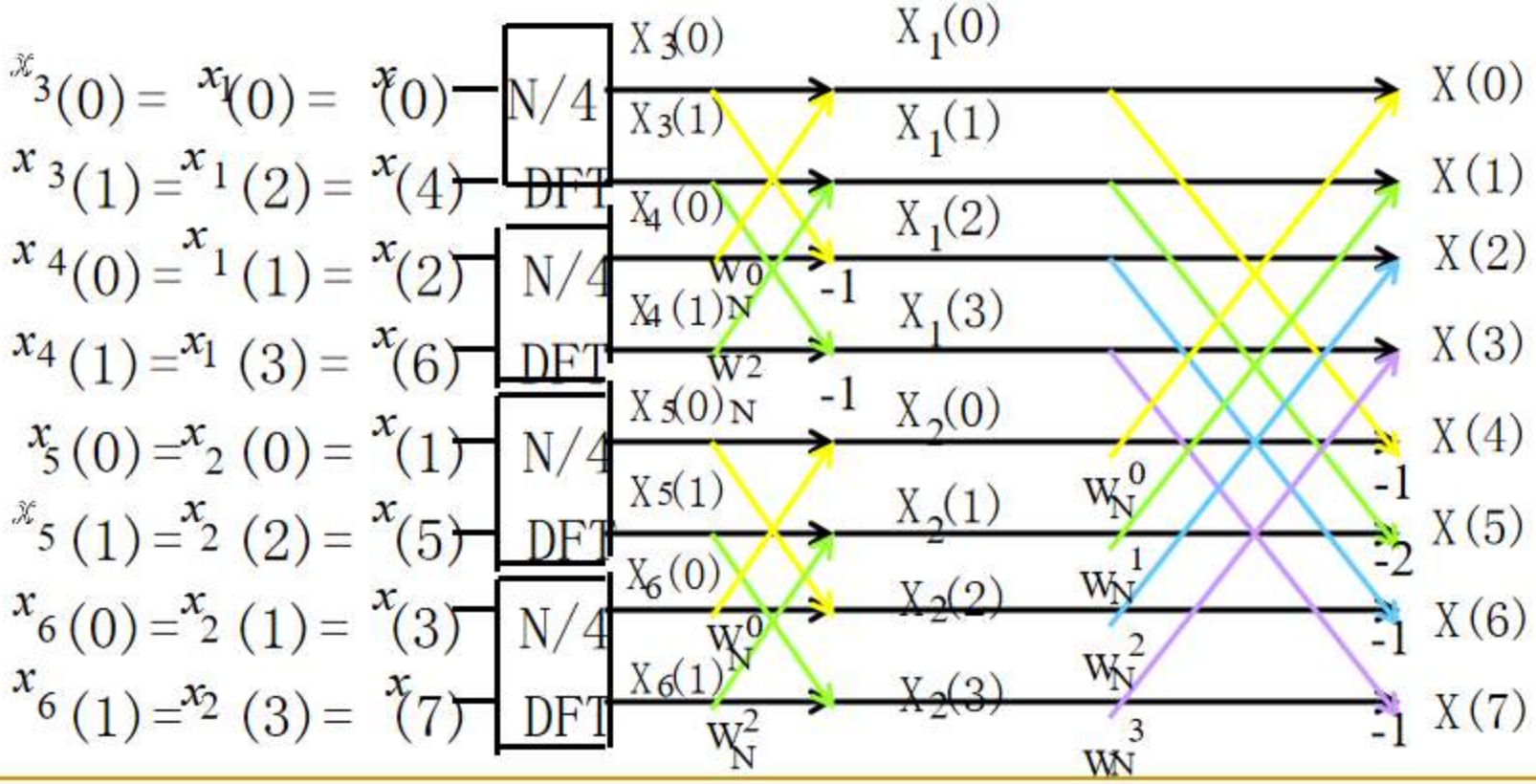
构成N/4点DFT，从而得到$X_6(0), X_6(1)$。

（5）由$X_3(0), X_3(1), X_4(0), X_4(1)$进行碟形运算，得到$X_1(0), X_1(1), X_1(2), X_1(3)$。

（6）由$X_5(0), X_5(1), X_6(0), X_6(1)$进行碟形运算，得到$X_2(0), X_2(1), X_2(2), X_2(3)$。

（7）由$X_1(0), X_1(1), X_1(2), X_1(3)$，$X_2(0), X_2(1), X_2(2), X_2(3)$进行碟形运算，得到

$X(0), X(1), X(2), X(3)\ X(4), X(5), X(6), X(7)$。



这样，又一次分解，得到四个N/4点DFT，两级蝶形运算，其运算量有大约减少一半即为N点DFT的1/4。

对于N=8时DFT，N/4点即为两点DFT，因此

$$X_3(k)=\sum_{l=0}^{1}x_3(l)W_{N/4}^{lk},\qquad k=0,1$$

$$X_4(k)=\sum_{l=0}^{1}x_4(l)W_{N/4}^{lk},\qquad k=0,1$$

$$X_5(k)=\sum_{l=0}^{1}x_5(l)W_{N/4}^{lk},\qquad k=0,1$$

$$X_6(k)=\sum_{l=0}^{1}x_6(l)W_{N/4}^{lk},\qquad k=0,1$$

亦即，

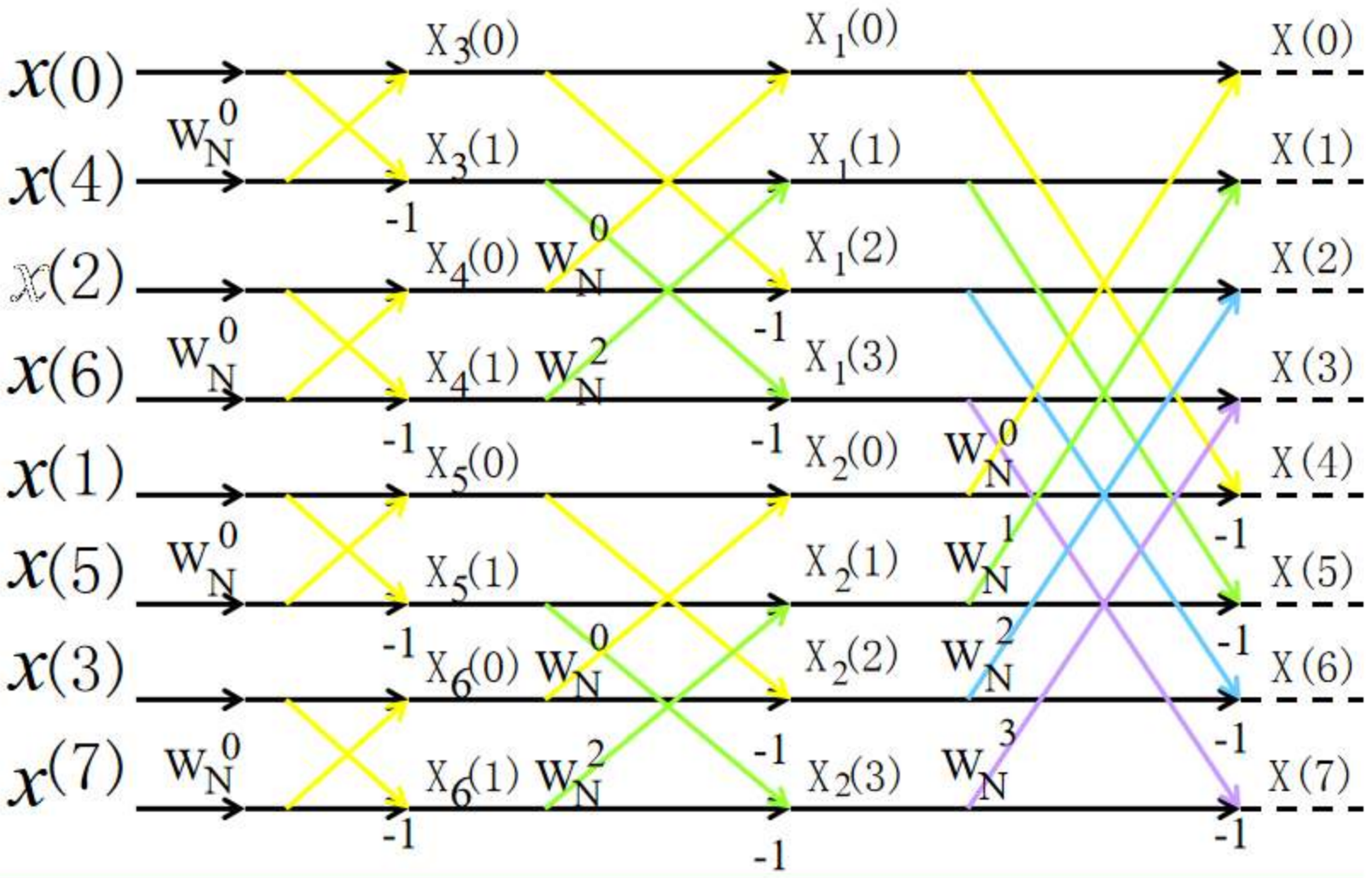
$$\begin{cases} X_3(0) = x_3(0) + W_2^0 x_3(1) = x(0) + W_N^0 x(4) \\ X_3(1) = x_3(0) + W_2^1 x_3(1) = x(0) - W_N^0 x(4) \end{cases}$$

$$\begin{cases} X_4(0) = x_4(0) + W_2^0 x_4(1) = x(2) + W_N^0 x(6) \\ X_4(1) = x_4(0) + W_2^1 x_4(1) = x(2) - W_N^0 x(6) \end{cases}$$

$$\begin{cases} X_5(0) = x_5(0) + W_2^0 x_5(1) = x(1) + W_N^0 x(5) \\ X_5(1) = x_5(0) + W_2^1 x_5(1) = x(1) - W_N^0 x(5) \end{cases}$$

$$\begin{cases} X_6(0) = x_6(0) + W_2^0 x_6(1) = x(3) + W_N^0 x(7) \\ X_6(1) = x_6(0) + W_2^1 x_6(1) = x(3) - W_N^0 x(7) \end{cases}$$

因此，8点DFT的FFT的运算流图如下



这种FFT算法，是在时间上对输入序列的次序是属于偶数还是属于奇数来进行分解的，所以称作按时间抽取的算法(DIT)。

二. 运算量

由上述分析可知，N=8需三级蝶形运算 $N=2^3=8$，由此可知，$N=2^L$ 共需L级蝶形运算，而且每级都由N/2个蝶形运算 组成，每个蝶形运算有一次复乘，两次复加。

因此，N点的FFT的运算量为

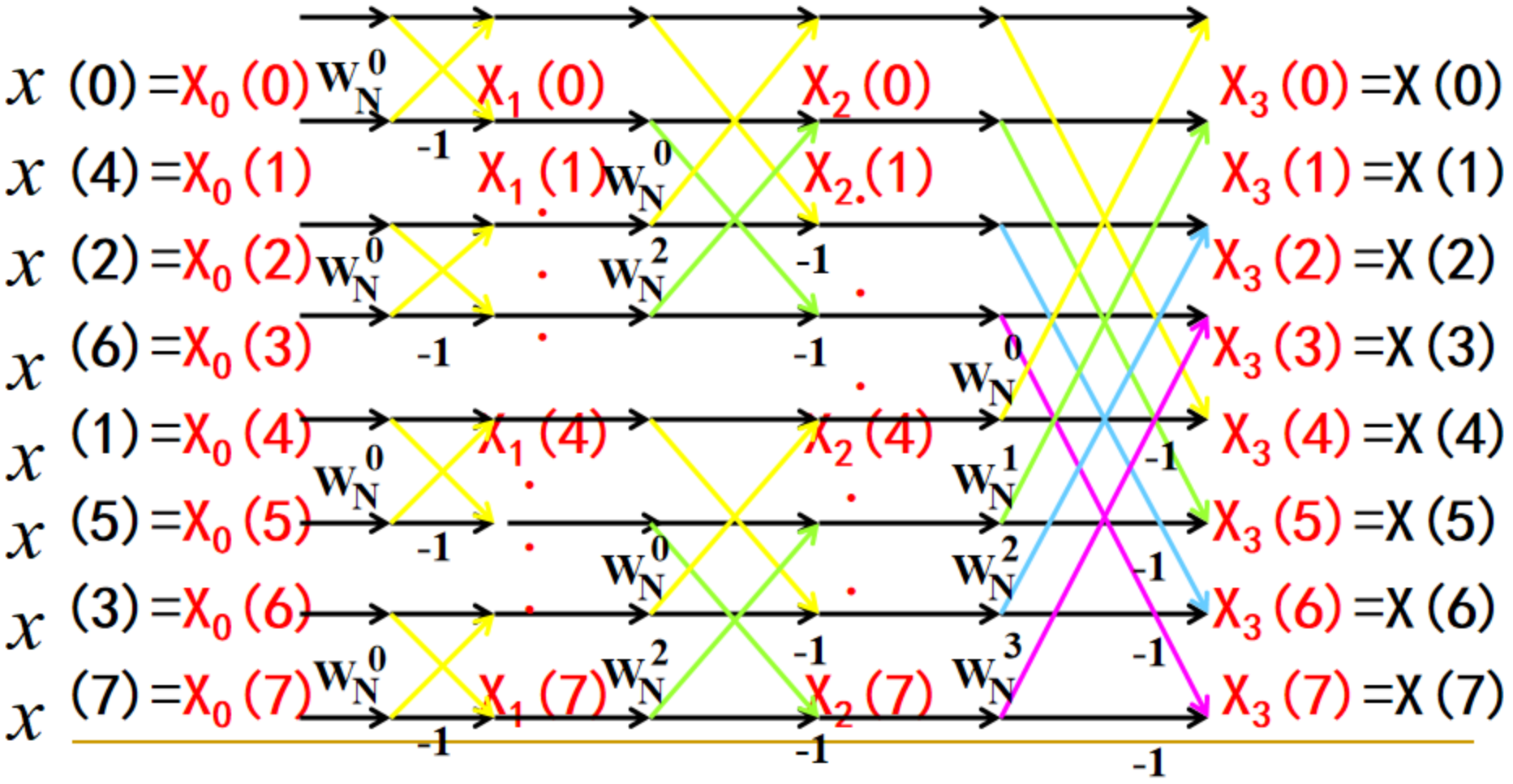
复乘：$m_F = (N/2)L = (N/2)\log_2 N$

复加：$a_F = N\ L = N\log_2 N$

三. DIT的FFT算法的特点

1. 原位运算

输入数据、中间运算结果和最后输出均用同一存储器。



由运算流图可知,一共有N个输入/出行,一共有$\log_2 N$=L列（级）蝶形运算(基本迭代运算).

设用m(m=1,2,… ,L)表示第m列;用k, j表示蝶形输入数据所在的（上/下）行数(0,1,2,… ,N-1);这时任何一个蝶形运算可用下面通用式表示，即

$$\begin{cases} X_m(k) = X_{m-1}(k) + X_{m-1}(j)W_N^r \\ X_m(j) = X_{m-1}(k) - X_{m-1}(j)W_N^r \end{cases}$$

$$x(0) = X_0(0), \qquad x(1) = X_0(4),$$
$$x(4) = X_0(1), \qquad x(5) = X_0(5),$$
$$x(2) = X_0(2), \qquad x(3) = X_0(6),$$
$$x(6) = X_0(3), \qquad x(7) = X_0(7).$$

所以，当m=1时，则有（前两个蝶形）

$$\begin{cases} X_1(0) = X_0(0) + X_0(1)W_N^0 \\ X_1(1) = X_0(0) - X_0(1)W_N^0 \end{cases}$$

$$\begin{cases} X_1(2) = X_0(2) + X_0(3)W_N^0 \\ X_1(3) = X_0(2) - X_0(3)W_N^0 \end{cases}$$

当m=2时,则有（前两个蝶形）

$$\begin{cases} X_2(0) = X_1(0) + X_1(2)W_N^0 \\ X_2(2) = X_1(0) - X_1(2)W_N^0 \end{cases}$$

$$\begin{cases} X_2(1) = X_1(1) + X_1(3)W_N^2 \\ X_2(3) = X_1(1) - X_1(3)W_N^2 \end{cases}$$

当m=3时,则有（前两个蝶形）

$$\begin{cases} X_3(0) = X_2(0) + X_2(4)W_N^0 \\ X_3(4) = X_2(0) - X_2(4)W_N^0 \end{cases}$$

$$\begin{cases} X_3(1) = X_2(1) + X_2(5)W_N^1 \\ X_3(5) = X_2(1) - X_2(5)W_N^1 \end{cases}$$

可见,在某列进行蝶形运算的任意两个节点(行)k和j的节点变量 $X_{m-1}(k), X_{m-1}(j)$ 就完全可以确定蝶形运算的结果 $X_m(k), X_m(j)$,与其它行(节点)无关。 这样,蝶形运算的两个输出值仍可放回蝶形运算的两个输入所在的存储器中,即实现所谓原位运算。每一组(列)有N/2个蝶形运算,所以只需N个存储单元,可以节 省存储单元。

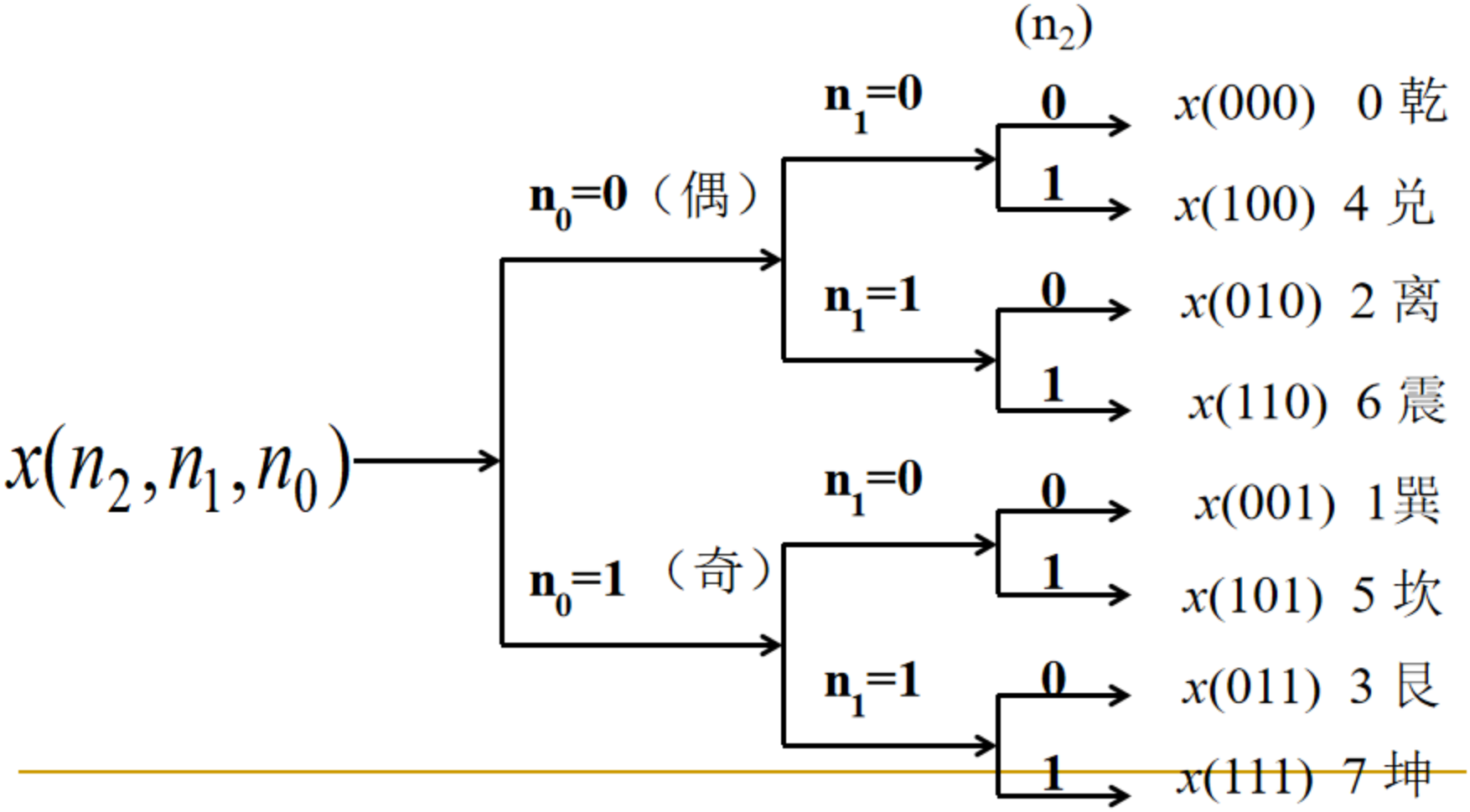
2 倒位序规律

由图可知, 输出$X(k)$按正常顺序排列在存储单元, 而输入是按顺序:

$$x(0), x(4), x(2), x(6)\text{；}\quad x(1), x(5), x(3), x(7)\text{；}$$

这种顺序称作倒位序, 即二进制数倒位。

这是由奇偶分组造成的，以N=8为例说明如下：



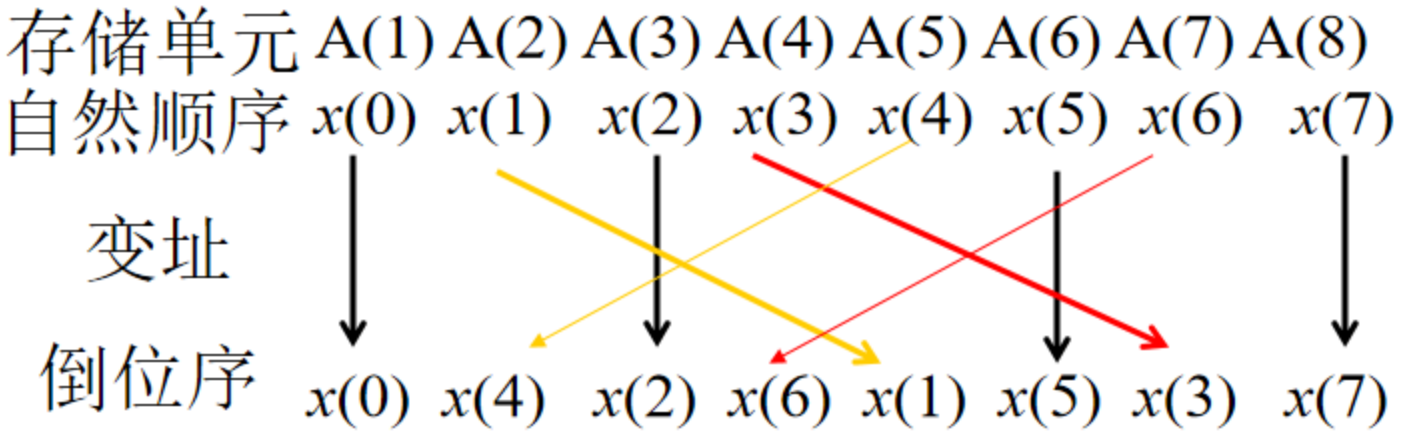
3. 倒位序实现

输入序列先按自然顺序存入存储单元，然后经变址运算来实现倒位序排列

设输入序列的序号为n，二进制为 $(n_2\ n_1\ n_0\ )_2$，倒位序顺序用$\hat{n}$表示，其倒位序二进制为 $(n_0\ n_1\ n_2\ )_2$，例如，N=8时如下表：

自然顺序n	二进制$n_2\ n_1\ n_0$	倒位序二进制$n_0\ n_1\ n_2$	倒位顺序n^
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

变址处理方法



4. 蝶形运算两节点的距离：2^{m-1}

其中，m表示第m列，且m =1，… ，L

例如$N=8=2^3$，第一级（列）距离为$2^{1-1}=1$，

第二级（列）距离为$2^{2-1}=2$，

第三级（列）距离为$2^{3-1}=4$。

5. W_N^r 的确定(仅给出方法)

考虑蝶形运算两节点的距离为2^{m-1}，蝶形运算可表为

$$\begin{cases} X_m(k)=X_{m-1}(k)+X_{m-1}(k+2^{m-1})W_N^r \\ X_m(k+2^{m-1})=X_{m-1}(k)-X_{m-1}(k+2^{m-1})W_N^r \end{cases}$$

由于N为已知，所以将r的值确定即可。

为此，令$k=(n_2n_1n_0)_2$，再将$k=(n_2n_1n_0)_2$左移$(L-m)$位，右边位置补零，就可得到$(r)_2$的值，即$(r)_2=(k)_2 2^{L-m}$。

例如 N=8=2^3

(1) $k=2$ ， $m=3$ 的r值, ∵$k=2=(010)_2$ 左移$L-m=3-3=0$ ，∴ $r=(010)_2=2$;

(2)$k=3$ ，$m=3$ 的r值； k= 3=$(011)_2$, 左移0位, r=3;

(3)k=5 ，m=2的值； k=5=$(101)_2$ 左移L-m=1位 r=$(010)_2$ =2。

6. 存储单元

存输入序列 $x(n)$, n=0, 1, , N−1, 计N个单元;

存放系数 W_N^r, r=0, 1, , (N/2) −1, 需N/2个存储单元;

共计(N+N/2)个存储单元。

§4-3 DIF的FFT算法(桑德—图基算法)

一. 算法原理

1. N点DFT的另一种表达形式

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

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

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

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

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

因此 $X(k)$可表为

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

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

2. N点DFT按k的奇偶分组可分为两个N/2的DFT

当k为偶数，即 k=2r时，$(-1)^k=1$；

当k为奇数，即 k=2r+1 时 $(-1)^k=-1$ 。

这时 $X(k)$ 可分为两部分：

k为偶数时：

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

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

k为奇数时：

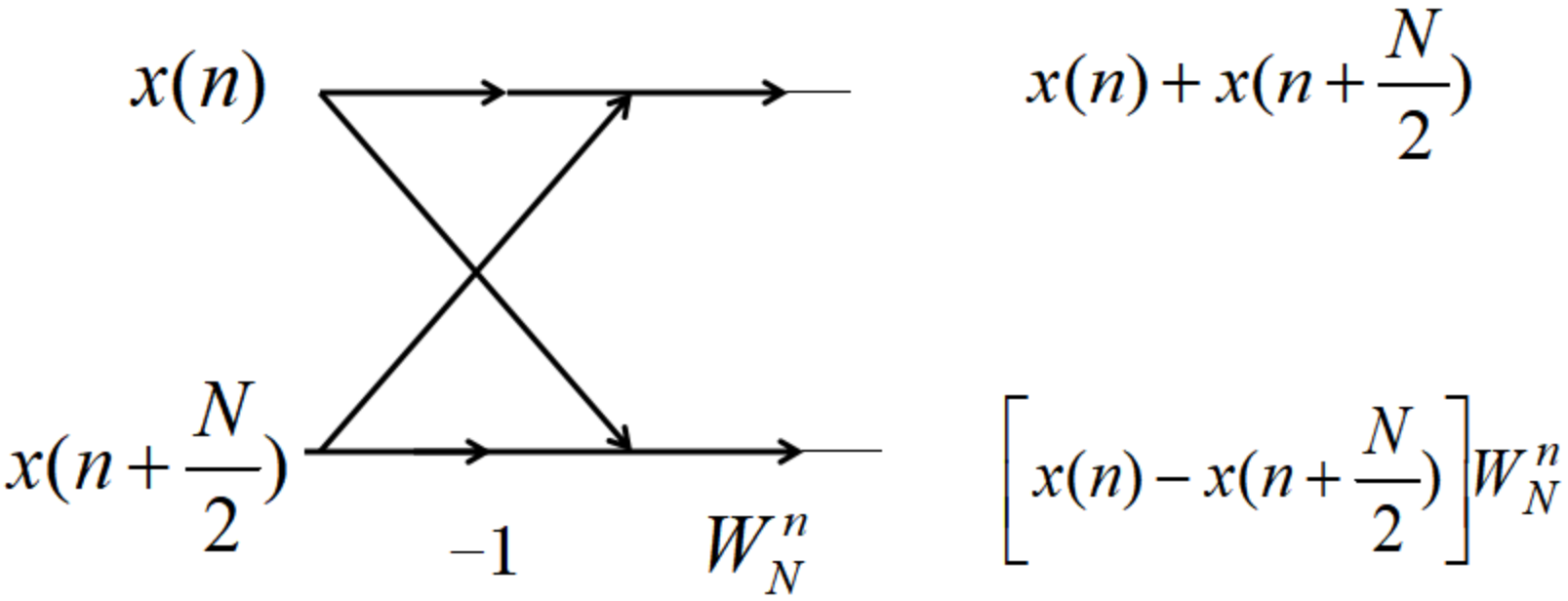
$$X(2r+1)=\sum_{n=0}^{\frac{N}{2}-1}\left[x(n)-x(n+\frac{N}{2})\right]W_N^{n(2r+1)}$$

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

可见，上面两式均为N/2的DFT。

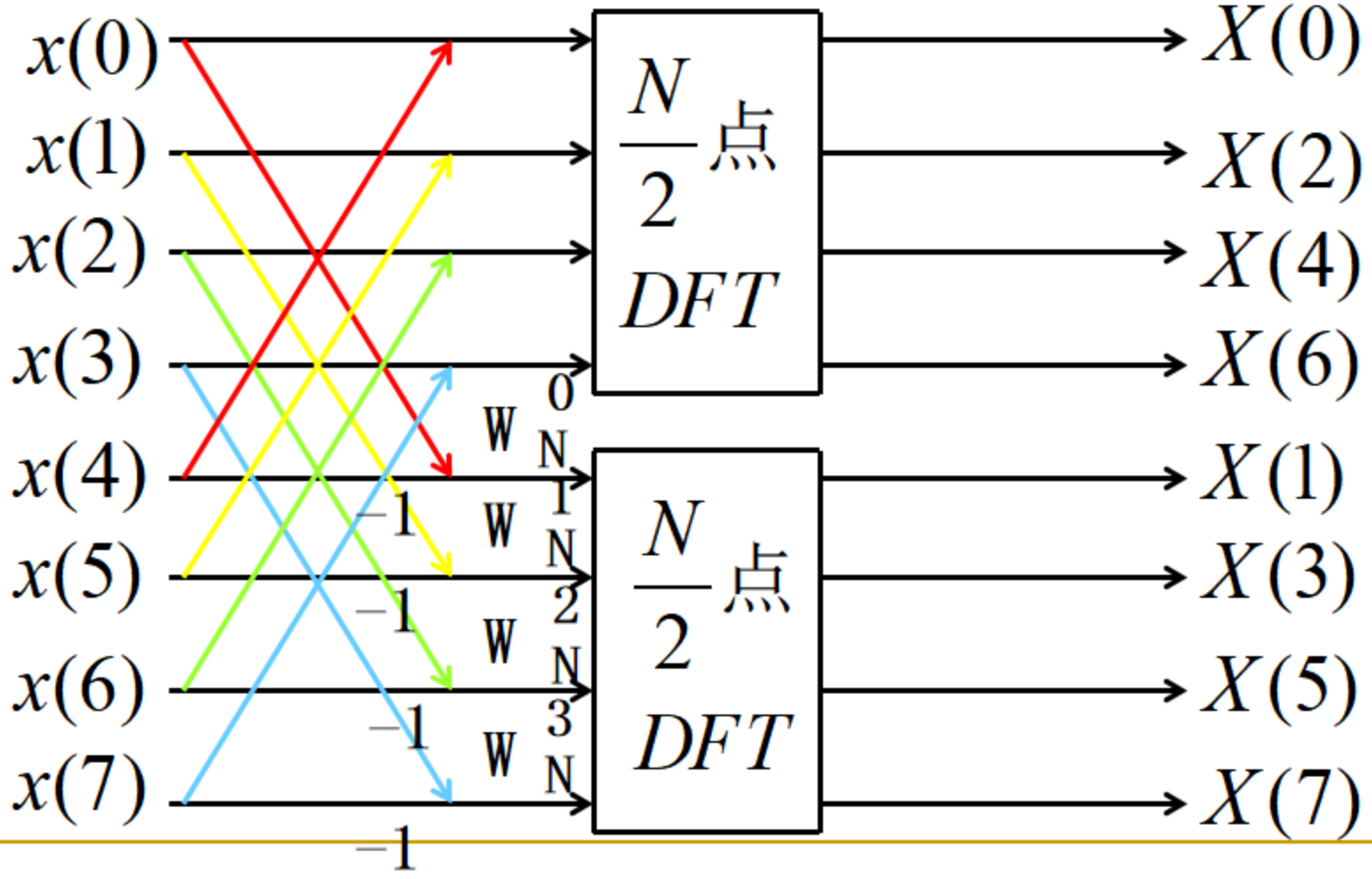
3. 蝶形运算

$x(n)$和$x(n+\frac{N}{2})$进行如下碟形运算：



4. N=8时，按k的奇偶分解过程

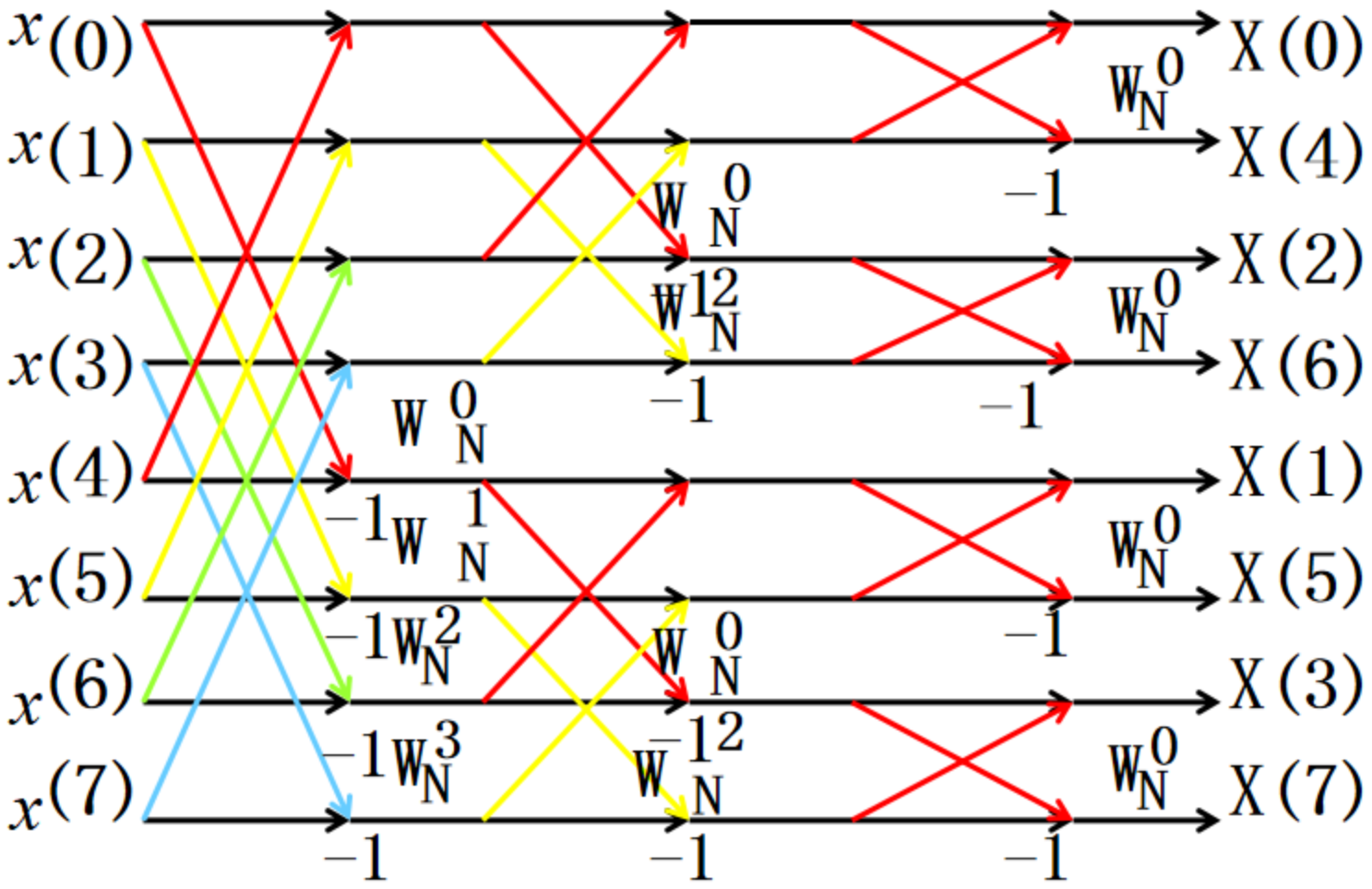
先蝶形运算，后DFT：



5. 仿照DIT的方法

再将N/2点DFT按k的奇偶分解为两个N/4点的DFT, 如此进行下去, 直至分解为2点DFT。

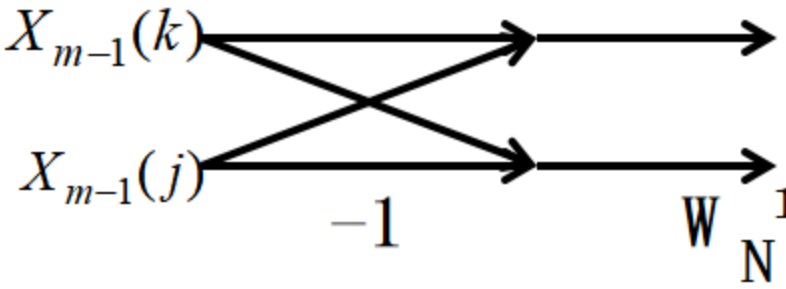
例如 N=8时DIF的FFT流图如下：



二.原位运算

每级(列)都是由N/2个蝶形运算构成,即

$$\begin{cases} X_m(k) = X_{m-1}(k) + X_{m-1}(j) \\ X_m(j) = [X_{m-1}(k) - X_{m-1}(j)]W_N^r \end{cases}$$



$$X_m(k) = X_{m-1}(k) + X_{m-1}(j)$$

$$X_m(j) = [X_{m-1}(k) - X_{m-1}(j)]W_N^r$$

三.蝶形运算两节点的距离

一般公式为$2^{L-m}=N/2^{m}$

例如 $N=2^{3}=8$ ：

(1)m=1 时的距离为 8/2=4;

(2)m=2 时的距离为 8/4=2;

(3)m=3 时的距离为 8/8=1。

四. W_N^r 的计算

由于DIF蝶形运算的两节点的距离为 $N/2^m$ ，所以蝶形运算可表为：

$$\begin{cases} X_m(k) = X_{m-1}(k) + X_{m-1}(k+\dfrac{N}{2^m}) \\ X_m(k+\dfrac{N}{2^m}) = [X_{m-1}(k) - X_{m-1}(k+\dfrac{N}{2^m})]W_N^r \end{cases}$$

r的求法：

$k=(n_2\ n_1\ n_0)$ ，左移m-1位，右边空出补零，得 $(r)_2$ ，亦即 $(r)_2 = (k)_2\ 2^{m-1}$.

例如，N=8：

(1) m=1, k=2, $k=(010)_2$ 左移0位, $(r)_2=(010)_2=2$;

(2) m=2, k=1, $k=(001)_2$ 左移1位, $(r)_2=(010)_2=2$;

(3) m=2, k=5, $k=(101)_2$ 左移1位, $(r)_2=(010)_2=2$.

五.DIF法与DIT法的异同

1. 相同点

(1) 进行原位运算；

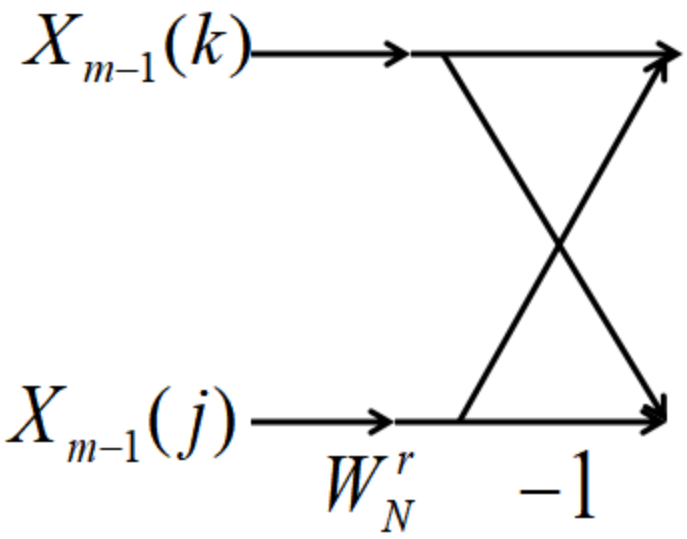
(2) 运算量相同，均为（N/2）Log_2N次复乘，N Log_2N次复加。

2. 不同点

(1) DIT输入为倒位序，输出为自然顺序；DIF正好与此相反。但DIT也有输入为自然顺序，输出为倒位序的情况。

(2) 蝶形运算不同

a. (DIT)



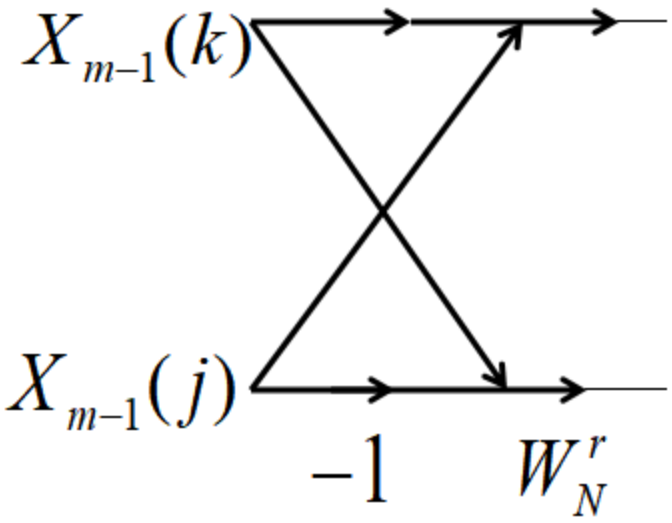
$$X_m(k) = X_{m-1}(k) + X_{m-1}(j)W_N^r$$

$$X_m(j) = X_{m-1}(k) - X_{m-1}(j)W_N^r$$

用矩阵表示

$$\begin{pmatrix} X_m(k) \\ X_m(j) \end{pmatrix} = \begin{pmatrix} 1 & W_N^r \\ 1 & -W_N^r \end{pmatrix} \begin{pmatrix} X_{m-1}(k) \\ X_{m-1}(j) \end{pmatrix}$$

b. (DIF)



$$X_m(k) = X_{m-1}(k) + X_{m-1}(j)$$

$$X_m(j) = [X_{m-1}(k) - X_{m-1}(j)]W_N^r$$

用矩阵表示

$$\begin{pmatrix} X_m(k) \\ X_m(j) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ W_N^r & -W_N^r \end{pmatrix} \begin{pmatrix} X_{m-1}(k) \\ X_{m-1}(j) \end{pmatrix}$$

(3) 两种蝶形运算的关系

互为转置（矩阵）；

a. (DIT)

$$\begin{pmatrix} 1 & W_N^r \\ 1 & -W_N^r \end{pmatrix}$$

b. (DIF)

$$\begin{pmatrix} 1 & 1 \\ W_N^r & -W_N^r \end{pmatrix}$$

将流图的所有支路方向都反向，交换输入和输出，即可得到另一种蝶形。

§4-4 IFFT算法

一.稍微变动FFT程序和参数可实现IFFT

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

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

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。

另外，可以将常数1/N分配到每级运算中，

$\because \quad \frac{1}{N}=\frac{1}{2^L}=(\frac{1}{2})^L$，也就是每级蝶形运算均乘以1/2。

二.不改(FFT)的程序直接实现IFFT

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

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

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

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

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

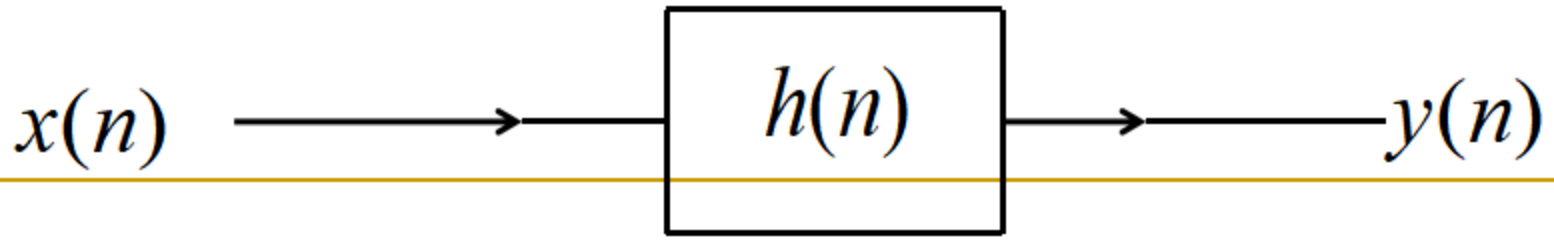
这就是说，先将X(k)取共轭，即将X(k)的虚部乘-1，直接利用FFT程序计算DFT；然后再取一次共轭；最后再乘1/N，即得χ(n)。所以FFT，IFFT可用一个子程序。

§4-5 线性卷积的FFT算法

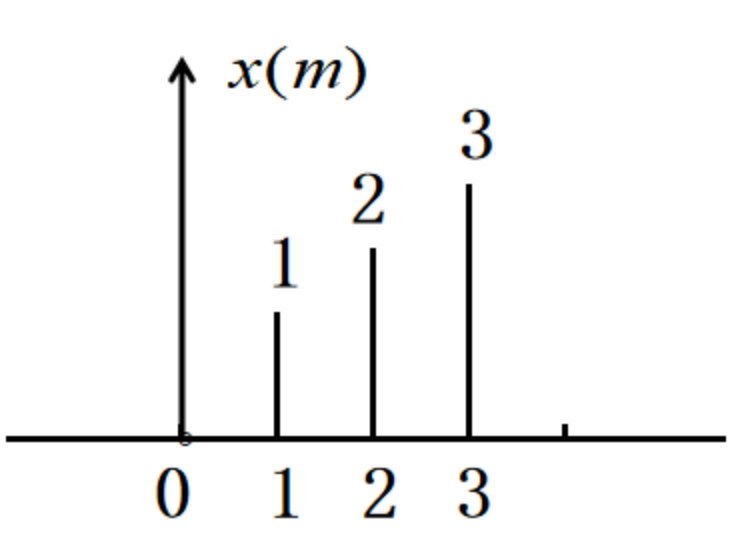
一. 线性卷积的长度

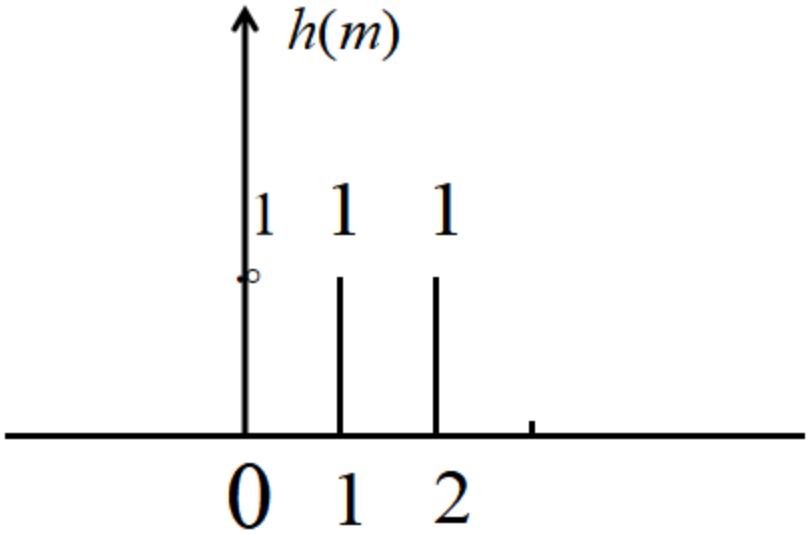
设一离散线性移不变系统的冲激响应为$h(n)$, 其输入信号为$x(n)$. 其输出为 $y(n)$. 并且$x(n)$的长度为L点, $h(n)$ 的长度为M点, 则：

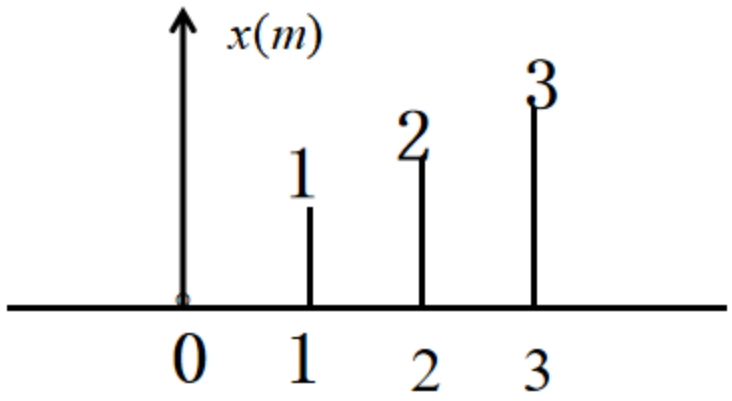
$$y(n)=x(n)*h(n)=\sum_{m=0}^{L-1}x(m)h(n-m)$$

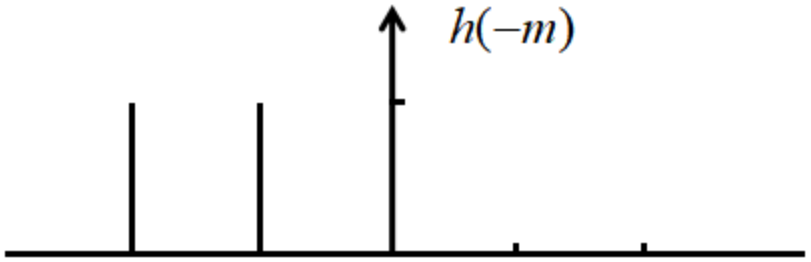


以实例说明：

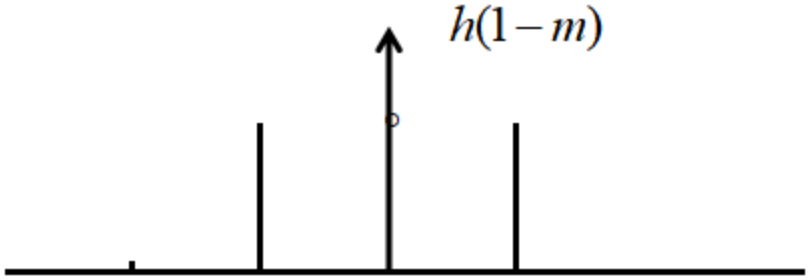




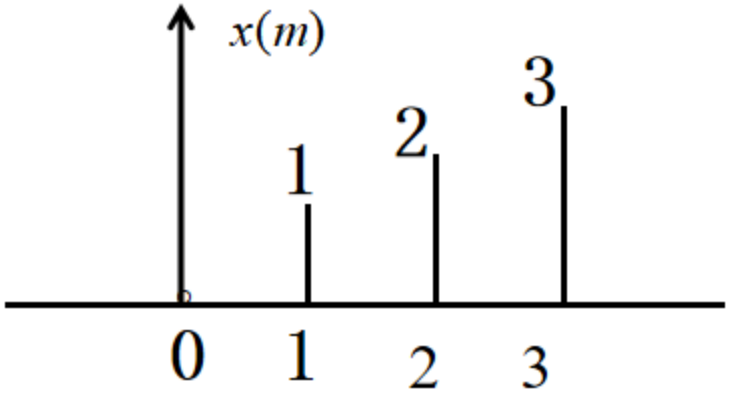


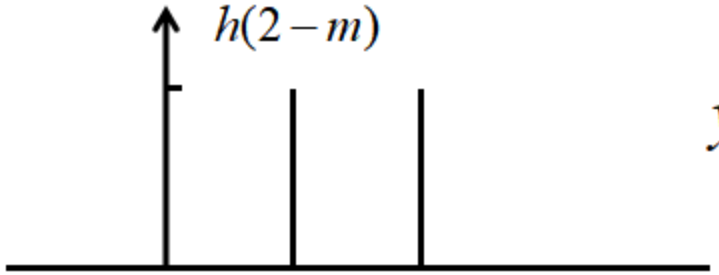


$$y(0) = \sum_{m=0}^{3} x(m)h(-m) = 0$$

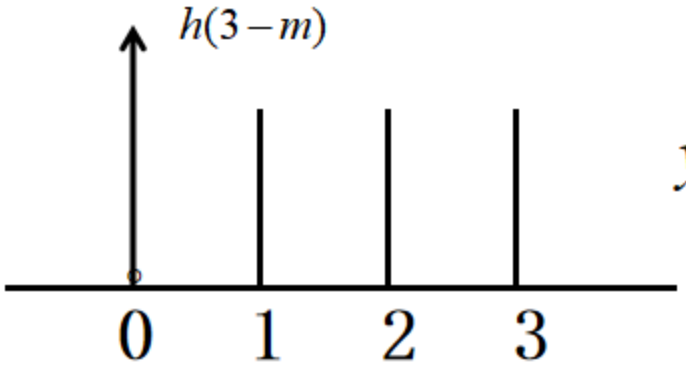


$$y(1) = \sum_{m=0}^{3} x(m)h(1-m) = 1$$

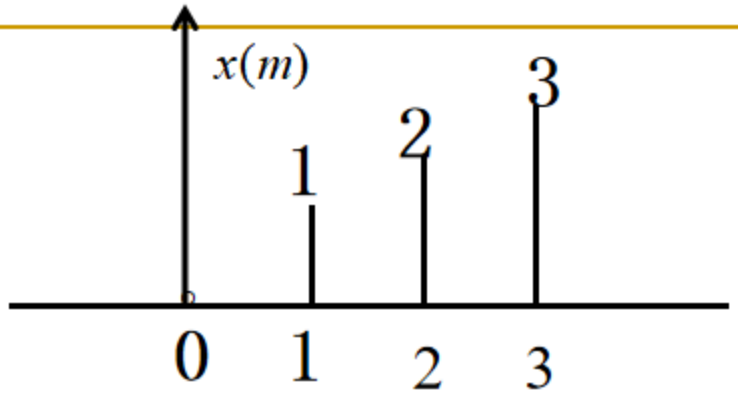


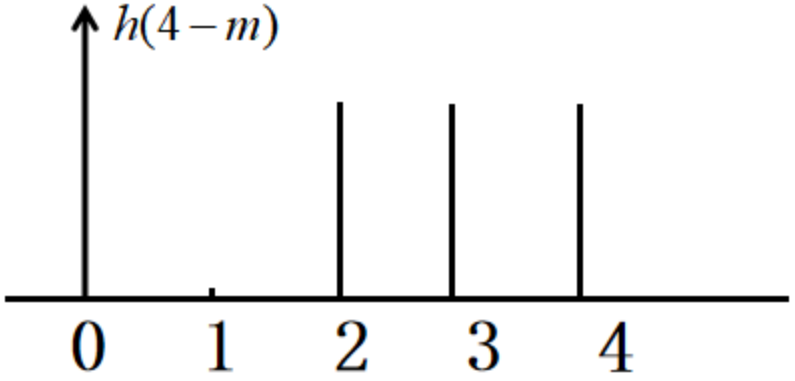


$$y(2) = \sum_{m=0}^{3} x(m)h(2-m) = 3$$

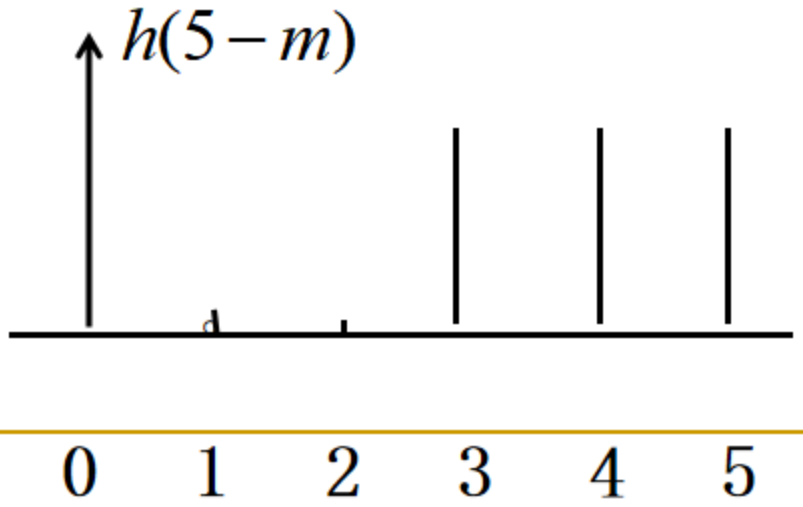


$$y(3) = \sum_{m=0}^{3} x(m)h(3-m) = 6$$

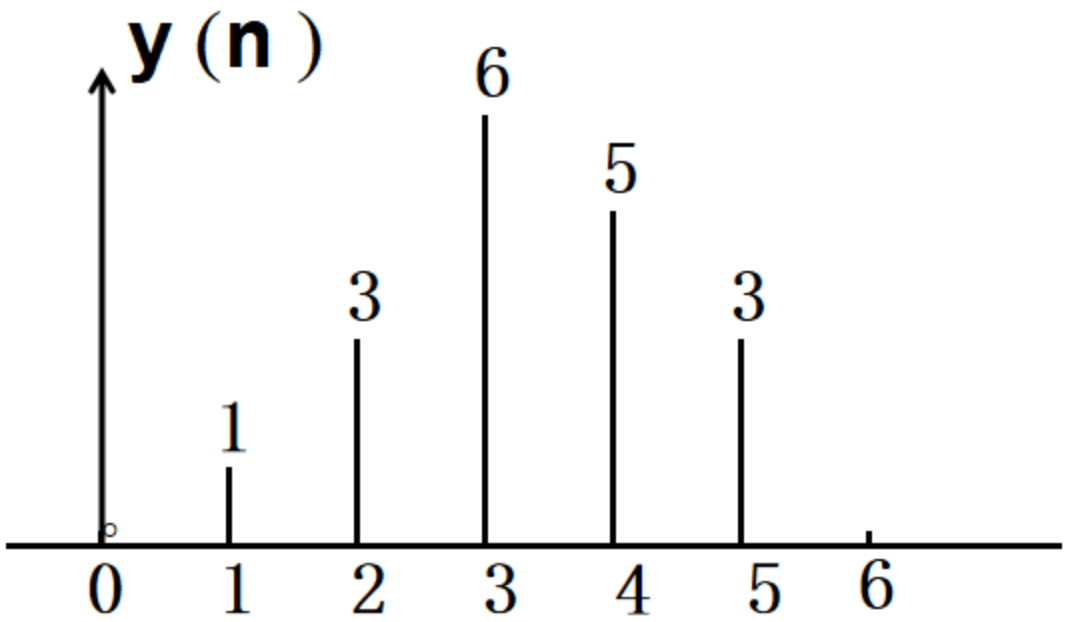




$$y(4) = \sum_{m=0}^{3} x(m)h(4-m) = 5$$



$$y(5) = \sum_{m=0}^{3} x(m)h(5-m) = 3$$



可见，$y(n)$的长度为$L+M-1$

二.用FFT算 y(n)的步骤：

1.将$x(n), h(n)$补零点，至少为$N = M + L - 1$点;

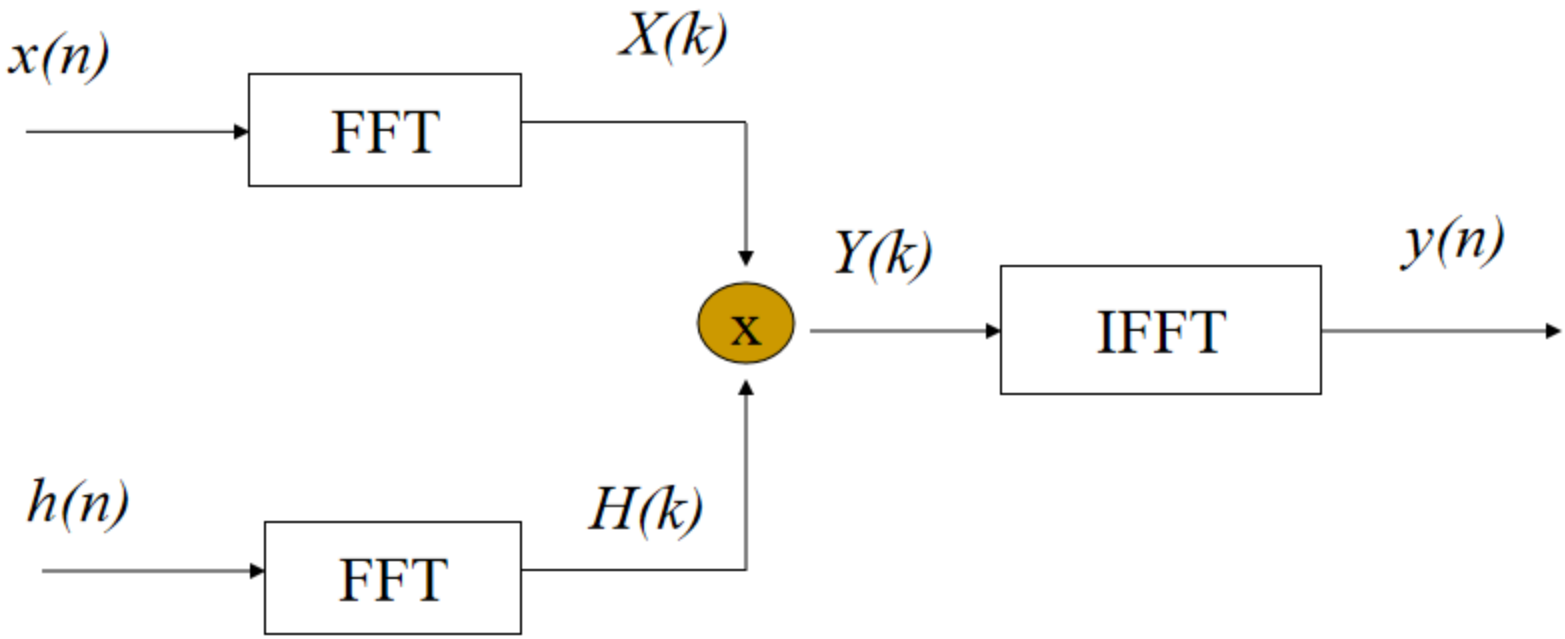
2.求$H(k) = FFT[h(n)]$;

3.求$X(k) = FFT[x(n)]$;

4.求$Y(k) = X(k)H(k)$;

5.求$y(n) = IFFT[Y(k)]$。

流程图



三.几点说明

1. 当 M=L 时,用圆周卷积计算线性卷积的速度快,点数越多,速度越快,N≥64时,速度增加明显.

2. L>>M 时,速度增加不太明显,为了增加速度,采用 (1)重叠相加法 (2)重叠保留法(从略).

放映结束