第四章 快速傅立叶变换(FFT)

4.1 引言

- FFT是DFT的一种快速算法
- 提出与发展
  - 由库利(J.K.Cooly) 和图基(J.K Tuky)
  - 相继出现了桑得(G.Sand)-图基等快速算法
- 价值
  - 使运算效率提高了1~2个数量级
  - 推动了数字信号处理技术的应用和发展

4.2 基2 FFT算法

4.2.1 直接计算DFT的问题及改进的方法

4.2.2 时域抽取法基2 FFT基本原理

4.2.3 DIT-FFT算法运算量

4.2.4 DIT-FFT的运算规律及编程思想

4.2.5 频域抽取法FFT（DIF-FFT)

4.2.6 IDFT的高效算法

4.2 基2 FFT算法

4.2.1 直接计算DFT的问题及改进的方法

- DFT的定义

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

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

两者形式类似，差别只在于的指数符号不同，及常数因子。

运算量是相同的

（1）正变换的运算量

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

因为 $x(n),W_N^{nk},X(k)$ 均为复数

每计算一个X(k)需要

N次复数乘法, (N-1)次复数加法

计算 N 点X(k),则需要

N^2次复数乘法，N(N-1)次复数加法

（2）减少运算量的途径

W_N^{nk} 具有如下特性：

对称性 $(W_N^{nk})^* = W_N^{-nk}$ $W_N^{-m} = W_N^{N-m}$

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

周期性 $W_N^{nk} = W_N^{(n+N)k} = e^{-j\frac{2\pi}{N}(n+N)k}$

利用这些特性：

1.使DFT运算中的有些项可以合并。

2.可将长序列的DFT分解为短序列的DFT。

4.2.2 时域抽取法基2 FFT基本原理

FFT的基本思想在于, 将原有的N点序列分成两个较短的序列,这两个序列的DFT组合起来,得出原序列的DFT。

FFT算法分为两大类

时域抽取法(DIT)

频域抽取法(DIF)

一、时域抽取法基本原理

设 $N = 2^M$ M为自然数

将长度为N的序列x(n)按n的奇偶分成两组

$$x_1(r) = x(2r), r = 0,1,\cdots(N/2-1)$$

$$x_2(r) = x(2r+1), r = 0,1,\cdots(N/2-1)$$

则x(n)的DFT为

$$X(k) = \sum_{n=\text{偶数}} x(n)W_N^{kn} + \sum_{n=\text{奇数}} x(n)W_N^{kn}$$

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

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

时域抽取法基本原理

由于 $W_N^{2kr} = e^{-j\frac{2\pi}{N}2kr} = e^{-j\frac{2\pi}{N/2}kr} = W_{N/2}^{kr}$

所以

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

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

式中，$X_1(k) = \sum_{r=0}^{N/2-1} x_1(r) W_{N/2}^{kr} = \sum_{r=0}^{N/2-1} x(2r) W_{N/2}^{kr}$

$$X_2(k) = \sum_{r=0}^{N/2-1} x_2(r) W_{N/2}^{kr} = \sum_{r=0}^{N/2-1} x(2r+1) W_{N/2}^{kr}$$

是x(2r)与x(2r+1)的N/2点DFT。

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

其中 $X_1(k)=DFT[x_1(r)]$ $X_2(k)=DFT[x_2(r)]$

上式可见，一个N点DFT已分解为两个N/2点的DFT $X_1(k)$与$X_2(k)$的组合。但得到的是X(k)的前一半项。

要用$X_1(k)$，$X_2(k)$表达全部的X(k)，必须应用旋转因子的周期性

$$W_{N/2}^{rk}=e^{-j\frac{2\pi}{N/2}rk}=e^{-j\frac{2\pi}{N/2}r(k+\frac{N}{2})}=W_{N/2}^{r(k+\frac{N}{2})}$$

时域抽取法基本原理

由于 $W_{N/2}^{rk} = W_{N/2}^{r(k+\frac{N}{2})}$

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

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

将下式自变量k变为k+N/2

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

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

X(k)的后半部分为：

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

再考虑到旋转因子的对称性

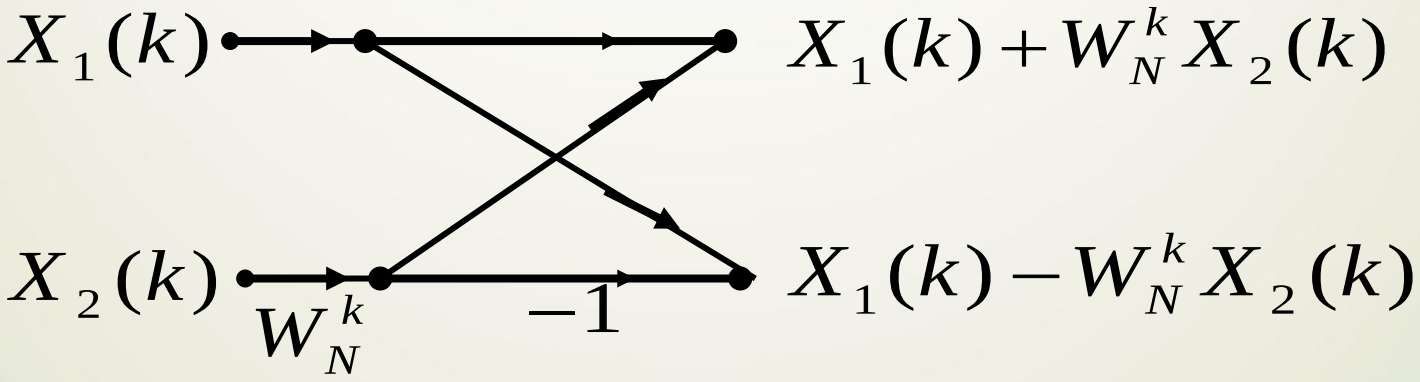
$$W_N^{k+\frac{N}{2}}=e^{-j\frac{2\pi}{N}(k+\frac{N}{2})}=e^{-j(\frac{2\pi k}{N}+\pi)}=-W_N^k$$

所以

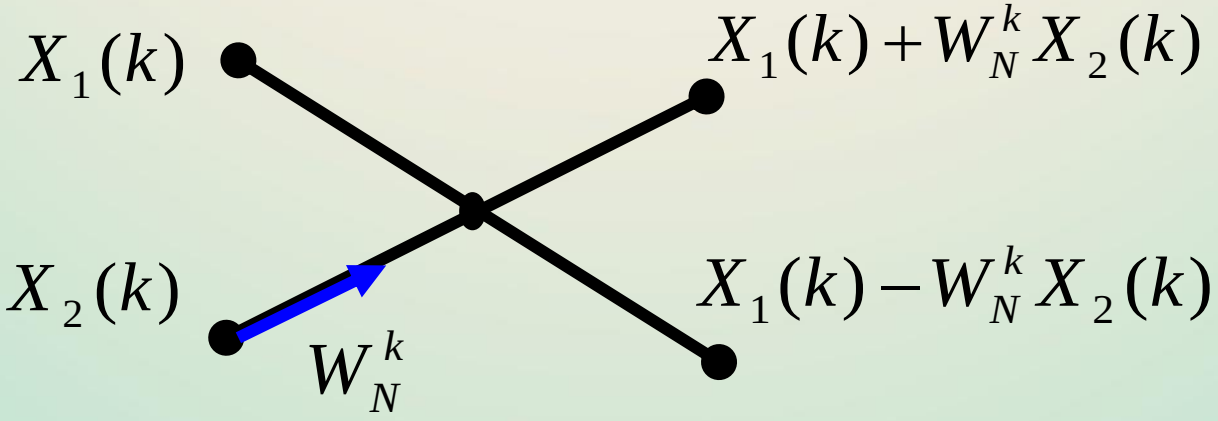
$$\begin{cases}X(k)=X_1(k)+W_N^kX_2(k)\\ X(k+\frac{N}{2})=X_1(k)-W_N^kX_2(k)\end{cases}\qquad 0\le k\le\frac{N}{2}-1$$

只要求出0～N/2-1区间上$X_1(k)$与$X_2(k)$的值，即可得到0～N-1区间内所有X(k)的值。

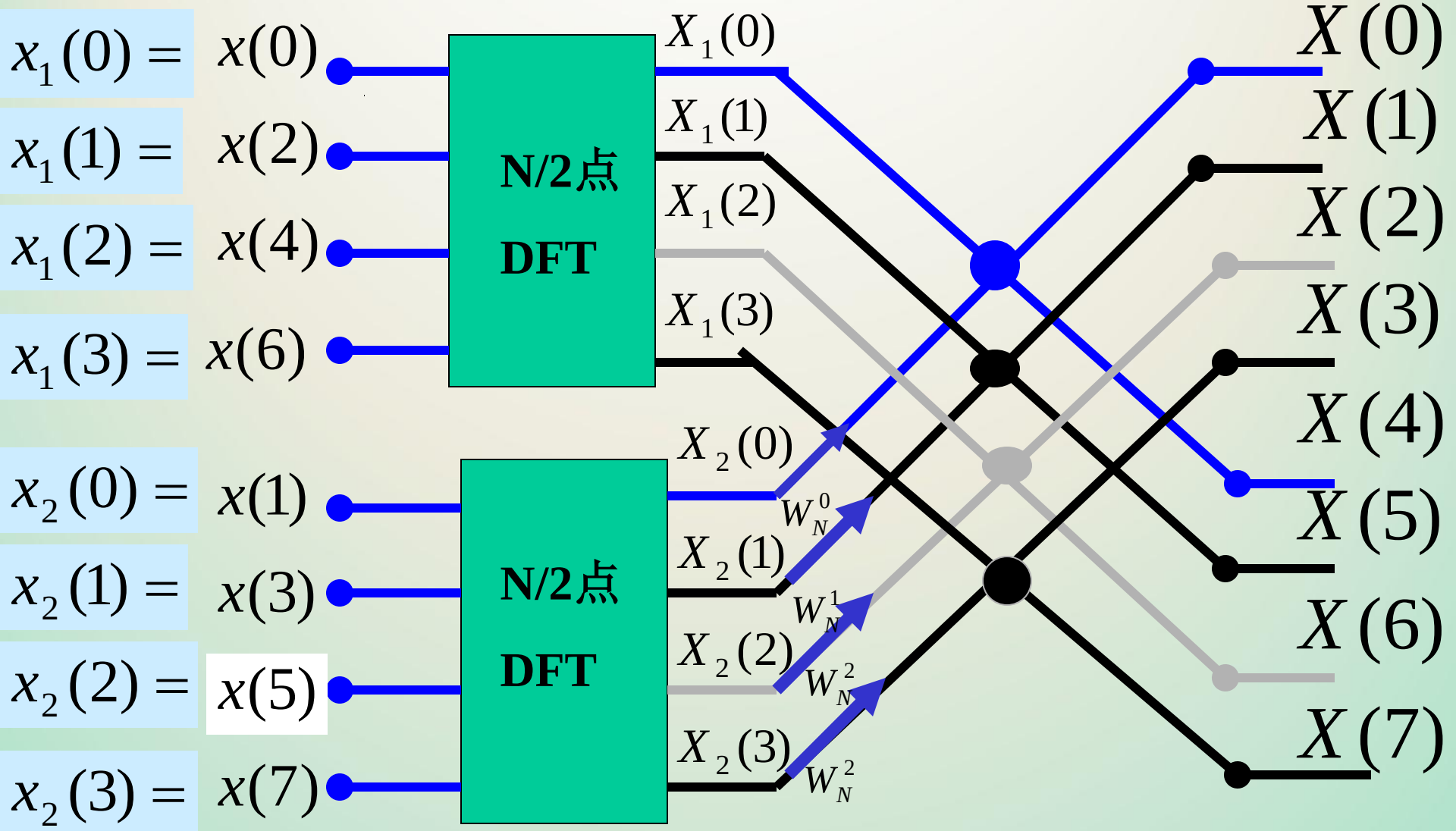
X(k)的运算可用蝶形信号流图表示



简记为下图形式：



N点DFT一次时域抽取分解图（N=8）



计算一个蝶形，需要1次复乘，2次复加。

每个N/2点的DFT需要$(N/2)^2$次复数乘，$N/2(N/2-1)$次复数加。

两个N/2点的DFT需要$N^2/2$次复数乘，$N(N/2-1)$次复数加。

将两个N/2点的DFT合并成N点DFT，有N/2个蝶形运算，还需要N/2次复数乘及N次复数加。

计算N点DFT共需要 $N^2/2+N/2\approx N^2/2$ 次复数乘，

$N(N/2-1)+N\approx N^2/2$次复数加。

只分解一次运算量就减少一半。

这种分解方法可一直进行到左侧为两点DFT为止。

与第一次分解相同，将$x_1(r)$按r的奇偶分成两个长为N/4的子序列：

$$\left.\begin{aligned} x_3(l) &= x_1(2l) \\ x_4(l) &= x_1(2l+1) \end{aligned}\right\}, l = 0,1,\cdots(N/4-1)$$

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

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

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

且 $$X_1(k+\frac{N}{4}) = X_3(k) - W_{N/2}^{k} X_4(k), \quad k = 0,1,\cdots\frac{N}{4}-1$$

$x_2(r)$ 也可以进行同样的处理，得到$X_2(k)$

$$X_2(k)=X_5(k)+W_{N/2}^{k}X_6(k),\quad k=0,1,\cdots\frac{N}{4}-1$$

$$X_2(k+\frac{N}{4})=X_5(k)-W_{N/2}^{k}X_6(k),\quad k=0,1,\cdots\frac{N}{4}-1$$

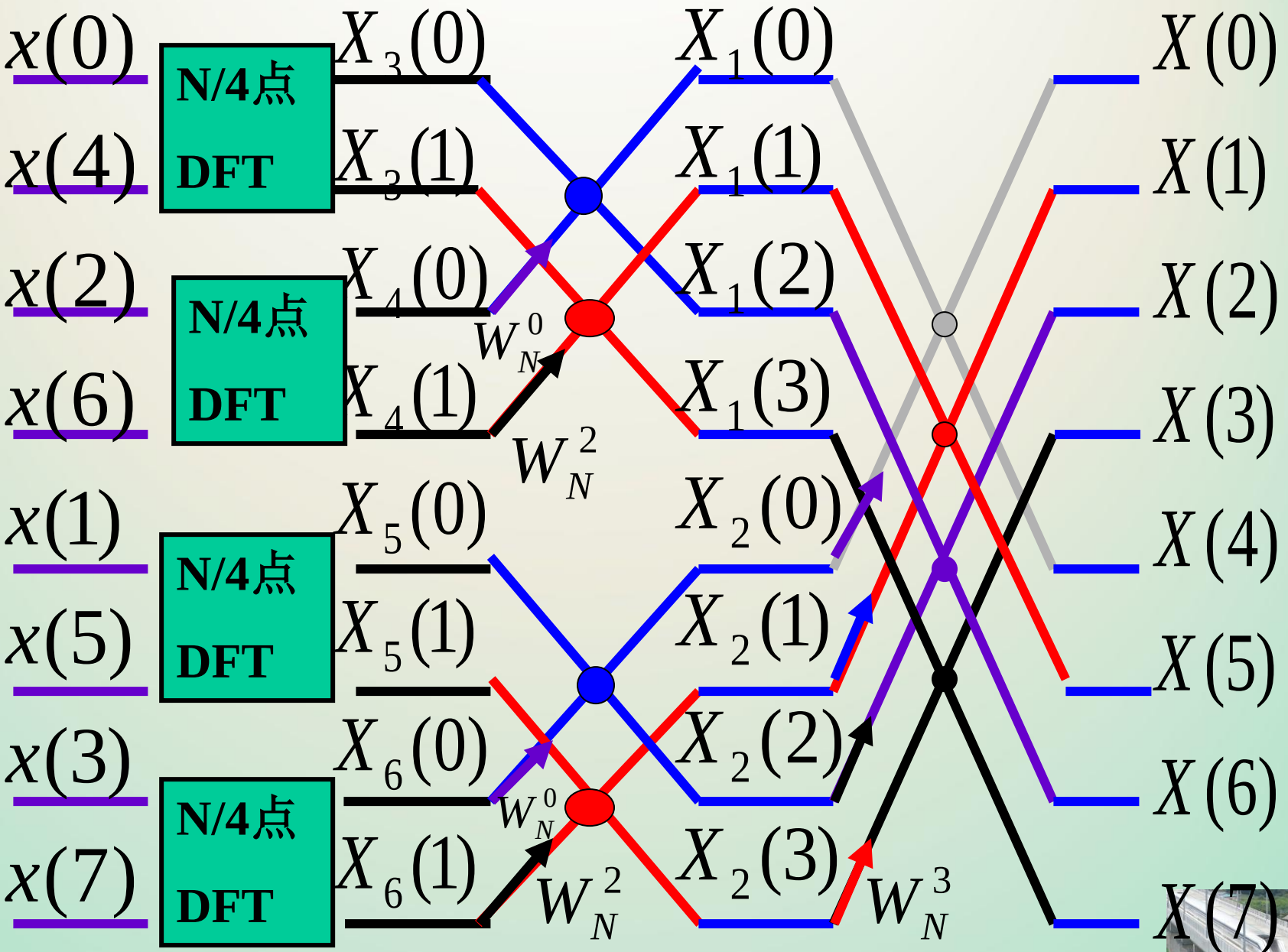
其中，$X_5(k)=\sum_{l=0}^{N/4-1}x_2(2l)W_{N/2}^{2kl}=\sum_{l=0}^{N/4-1}x_5(l)W_{N/4}^{kl}$

$$X_6(k)=\sum_{l=0}^{N/4-1}x_2(2l+1)W_{N/2}^{2kl}=\sum_{l=0}^{N/4-1}x_6(l)W_{N/4}^{kl}$$

将旋转因子统一为 $W_{N/2}^{k}=W_N^{2k}$ ，则

一个N点DFT就可分解为4个N/4点的DFT.

N点DIT-FFT运算流图



最后剩下的是2点DFT，当N=8时，其输出为：

$$X_3(k), X_4(k), X_5(k), X_6(k) \quad k=0,1$$

以$X_4(k)$为例，

$$X_4(k)=\sum_{l=0}^{N/4-1} x_4(l)W_{N/4}^{kl}=\sum_{l=0}^{1} x_4(l)W_2^{kl}\ ,k=0,1$$

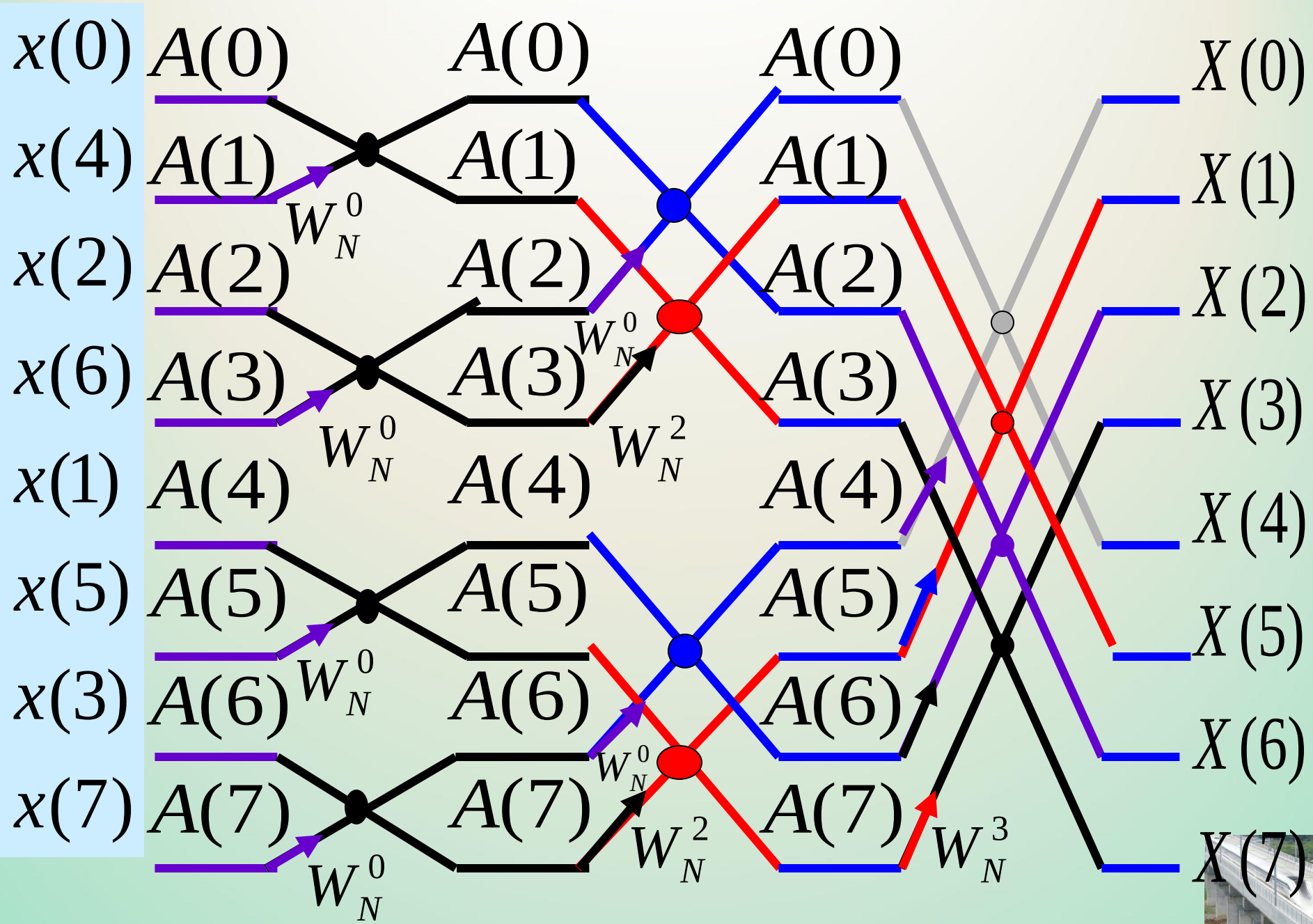
因为 $$W_2^1=e^{-j\frac{2\pi}{2}}=-1=-W_N^0$$

$$X_4(0)=x_4(0)+W_2^0x_4(1)=x(2)+W_N^0x(6)$$

$$X_4(1)=x_4(0)+W_2^1x_4(1)=x(2)-W_N^0x(6)$$

即x(2),x(6)通过蝶形运算得到$X_4(k)$

N点DIT-FFT运算流图



4.2.3 DIT-FFT算法运算量

当 $N = 2^M$ 时，可分解为M级蝶形，每级都有N/2个蝶形运算。

每一级 N / 2次复数乘；

N 次复数加。

则 M级 $\frac{N}{2} \bullet M = \frac{N}{2}\log_2 N$ 次复数乘

$N \bullet M = N\log_2 N$ 次复数加

与直接计算DFT的运算量之比

$$N^2 /(N/2)\log_2 N$$

4.2.4 DIT-FFT的运算规律及编程思想

1、原位计算（就地算法）

2、旋转因子的变化规律

3、蝶形运算规律及编程思想

4、倒位序规律

5、倒位序实现

1. 原位计算（就地算法）

用同一地址既存输入序列又存输出序列的算法。

如图4.2.4，每级运算由N/2个蝶形构成，每个蝶形完成下述基本运算：

$$X_L(J) = X_{L-1}(J) + X_{L-1}(J+B)W_N^p$$

$$X_L(J+B) = X_{L-1}(J) - X_{L-1}(J+B)W_N^p$$

式中L代表第L级蝶形运算。

J、J+B代表数据所在行。

B表示蝶形运算的两个输入相距B个点。

运算规律

每个蝶形运算的两个输入数据只对计算本蝶形有用，而且每个蝶形的输入、输出数据节点又在同一水平线上。

这样，蝶形的两个输出值仍放回蝶形的两个输入所在的存储器中。

每列的**N/2**个蝶形运算全部完成后，再开始下一列的蝶形运算。

下一列仍采用原位运算，只是进入蝶形的组合关系有所不同。

2. 旋转因子 W_N^P 的变化规律

$$\begin{cases} X_L(J) = X_{L-1}(J) + X_{L-1}(J+B)W_N^p \\ X_L(J+B) = X_{L-1}(J) - X_{L-1}(J+B)W_N^p \end{cases}$$

W_N^p 称为旋转因子，p称为旋转因子的指数。

级(L):从左到右的运算级数。(L=1,2,…M)

观察图4.2.4，第L级共有2^{L-1}个不同的旋转因子。

旋转因子与级数（L）的关系

$$L=1 \quad W_N^P = W_{N/4}^J = W_{2^L}^J, J=0$$

$$L=2 \quad W_N^P = W_{N/2}^J = W_{2^L}^J, J=0,1$$

$$L=3 \quad W_N^P = W_N^J = W_{2^L}^J, J=0,1,2,3$$

更一般地 $N=2^M$ 第 **L** 级的旋转因子为

$$W_N^p = W_{2^L}^J = W_{N\cdot 2^{L-M}}^J = W_N^{J\cdot 2^{M-L}}, 0 \le J \le (2^{L-1}-1)$$

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

3. 蝶形运算规律

蝶形运算两输入点间距离为：

第1级 ： 1　　第2级 ： 2

第3级 ： 4　　第L级 ： 2^{L-1}

每一级的两个输入节点进行蝶形运算后，得到下一级的相同序号的两个输出节点。

$$X_L(J) = X_{L-1}(J) + X_{L-1}(J+2^{L-1})W_N^p$$

$$X_L(J+2^{L-1}) = X_{L-1}(J) - X_{L-1}(J+2^{L-1})W_N^p$$

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

$$X_L(J) = X_{L-1}(J) + X_{L-1}(J+2^{L-1})W_N^p$$

$$X_L(J+2^{L-1}) = X_{L-1}(J) - X_{L-1}(J+2^{L-1})W_N^p$$

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

对于每个旋转因子，有2^{M-L}个蝶形运算。

第一个蝶形的第一个输入所在行为J，

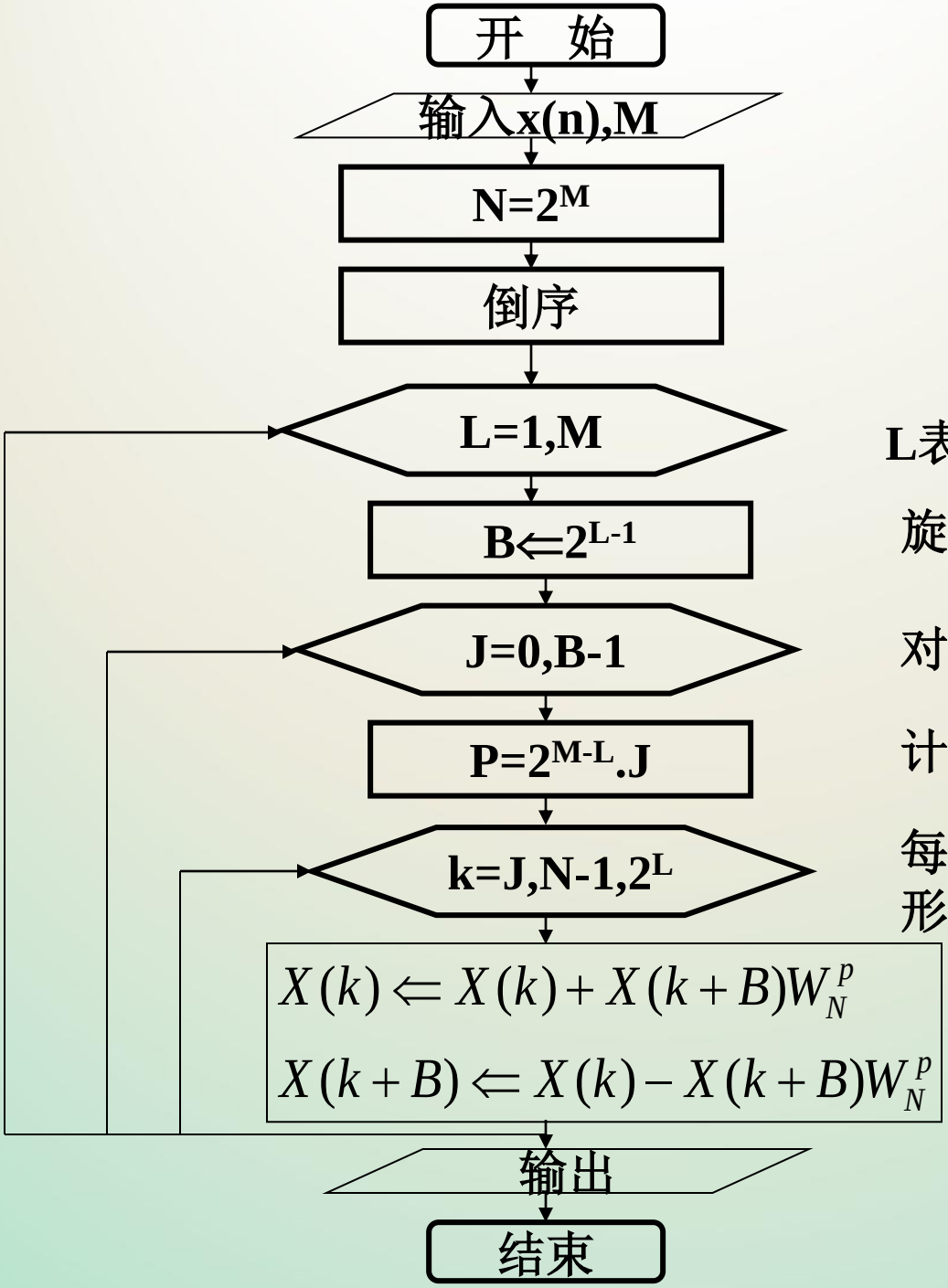
第二个蝶形的第一个输入所在行为$J+2^L$，

相邻两个蝶形运算第一个输入相距2^L。

编程思想

先从输入端开始，逐级进行计算，共进行M级运算。

在进行第L级运算时，依次求出2^{L-1}个不同的旋转因子，每求一个旋转因子，就计算完它对应的所有2^{M-L}个蝶形。



L表示运算级数

旋转因子个数

对旋转因子计数

计算旋转因子指数

每个旋转因子对应2^{M-L}个蝶形运算。

两个蝶形运算第一个输入点‘距离’是2^L

4. 倒位序规律

若 $n_0 n_1 n_2$ 是三位二进制数，

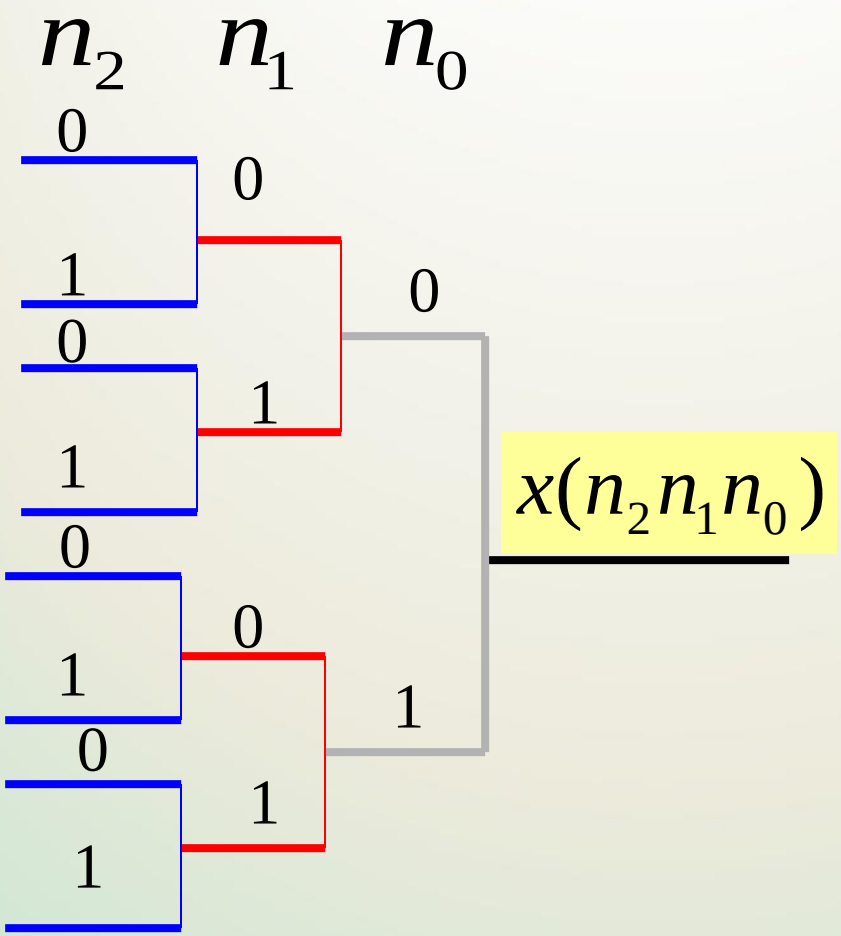
则 $n_2 n_1 n_0$ 就是它的倒位序。

按原位计算时，FFT的输出X(k)是按自然顺序存储的，但这时输入序列却不是按自然顺序存储的。

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

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

输入序列初看起来，好象没有规律，实际是按倒位序存储的。



倒位序的形成

x(000)=x(0)

x(100)=x(4)

x(010)=x(2)

x(110)=x(6)

x(001)=x(1)

x(101)=x(5)

x(011)=x(3)

x(111)=x(7)

造成倒位序的原因是输入序列x(n)，
按标号n的**奇偶**不断地分组造成的。

5. 倒位序的实现

（1）只要将顺序二进制数（$n_2n_1n_0$）的二进制位倒置，得（ $n_0n_1n_2$）。根据这种规律，容易用硬件电路和汇编语言产生倒位序数。

（2）用高级语言程序实现时，必须找出产生倒序数的十进制运算规律。

顺序与倒序二进制对照表

左边为按自然顺序排列的二进制数，下面的一个数是上面一个数的最低位上加上1，且向高位进位。

0	000	0	000
1	001	4	100
2	010	2	010
3	011	6	110
4	100	1	001
5	101	5	101
6	110	3	011
7	111	7	111

右边为倒位序数，下面的一个数是上面一个数的最高位上加上1，且由高位向低位进位。称为反向进位加法

可由当前任一倒序值求得下一个倒序值

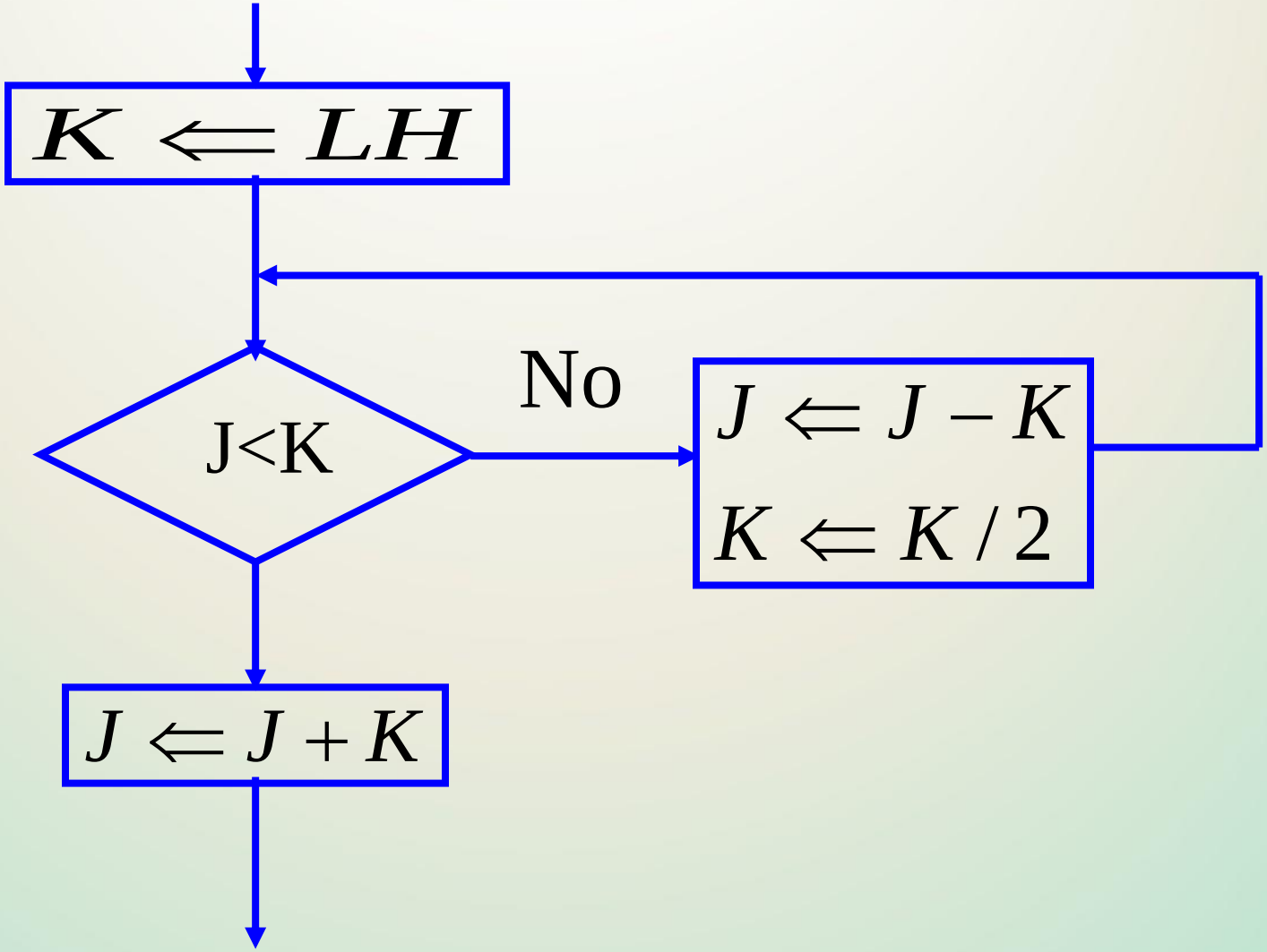
反向进位加法的实现

若已知某个倒位序数J，求下一个倒位序数，

- 判 断 J 的最高位是否为 “0”,
  - 让J与N/2比较，因为N/2总是100……,

    如果 J<N/2 ,则 J 的最高位为零,只需把该位变为 1(J+N/2),就得到下一个倒位序数,否则,把最高位变为 0(J-N/2)
- 判 断 J 的次高位是否为 “0”,
  - 让J与N/4比较，

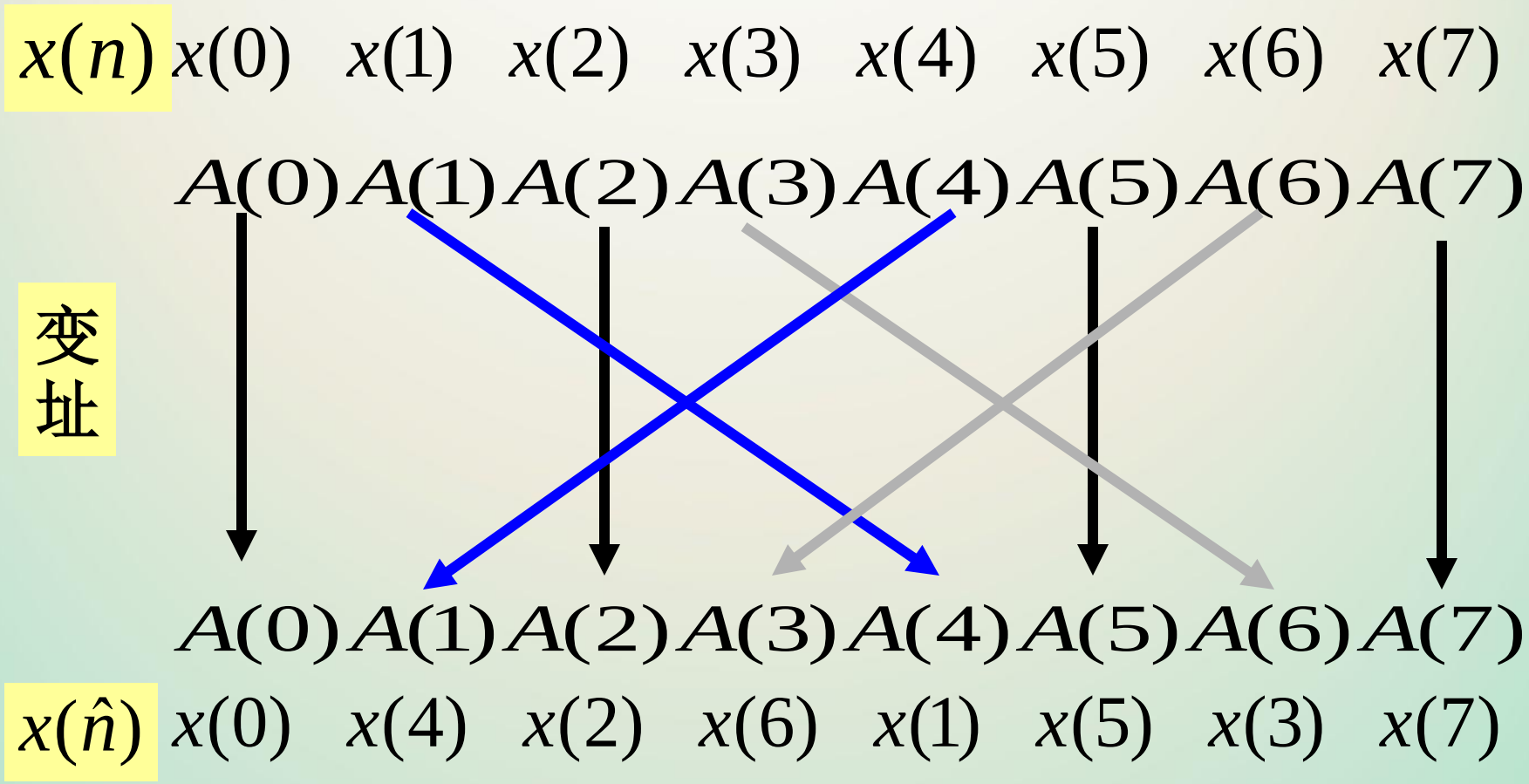
如果 J 的次高位为零,只需把该位变为 1(J+N/4),其它位不变,就得到下一个倒位序数,否则,还需判断下一位(与 N/8比较),如此依次进行下去,总会碰到某位为0,将此位改为1即可.


$K \Leftarrow LH$
J<K
No
$J \Leftarrow J - K$
$K \Leftarrow K / 2$
$J \Leftarrow J + K$

实现倒位序的流程图

形成倒序数后，把原来按自然顺序存放在存储单元的输入序列**x(n)**重新按倒序排列。

通过变址运算完成倒位序



$n = \hat{n}$ 时，不必调换。而当 $n < \hat{n}$ 时，需调换。

4.2.5 频域抽取法FFT（DIF-FFT)

与DIT相对应，DIF算法是将频域X(k)的序号k按奇偶分开。

推导过程

设 $N=2^M$ M为自然数

则x(n)的DFT为

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

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

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

4.2.5 频域抽取法FFT（DIF-FFT)

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

式中

$$W_N^{\frac{N}{2}k}=e^{-j\frac{2\pi}{N}\cdot\frac{N}{2}k}=(-1)^k=\begin{cases}1 & if\ \ k\ \ is\ \ even\\ -1 & if\ \ k\ \ is\ \ odd\end{cases}$$

分别令k=2r,k=2r+1,r=0.1……,N/2-1

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

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

4.2.5 频域抽取法FFT（DIF-FFT)

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

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

令

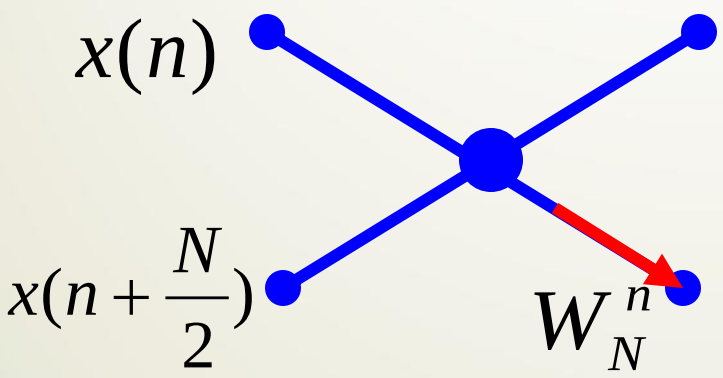
$$x_1(n)=\left[x(n)+x(n+\frac{N}{2})\right]$$

则

$$x_2(n)=\left[x(n)-x(n+\frac{N}{2})\right]W_N^n,0\le n\le N/2-1$$

$$X(2r)=\sum_{n=0}^{N/2-1}x_1(n)W_{N/2}^{nr}\quad X(2r+1)=\sum_{n=0}^{N/2-1}x_2(n)W_{N/2}^{nr}$$

4.2.5 频域抽取法FFT（DIF-FFT)



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

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

由于$N=2^M$，N/2仍然是偶数，继续将N/2点的DFT分成偶数组和奇数组。

这样每个N/2点DFT可由两个N/4点DFT形成。

其输入序列分别是$x_1(n)$和$x_2(n)$按上下对半分开形成的四个子序列。

这样将X(k)按k的奇偶把一个N点的DFT分成为两个N/2点的DFT,且可一直分解为直到两点的DFT.

- DIF与DIT算法的比较
  - 基本蝶形不同
  - 都有M级运算,每级有N/2个蝶形
  - 都可进行原位计算
  - 运算次数相同
  - 输入为自然顺序,输出为倒位序

4.2.6 IDFT的高效算法

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

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

比较两个公式可以看出，只要改变DFT运算中的每个系数符号，最后乘以 1/N ，就是IDFT的运算公式了。

实际中，为了避免发生溢出，将**1/N**分配到每一级蝶形运算中，因为 $1/N=(1/2)^M$ 所以每一级的每个蝶形输出到乘以**1/2.**

4.2.6 IDFT的高效算法

- DITFFT用于计算IFFT时，称为DIT-IFFT
- DIFFFT用于计算IFFT时，称为DIF-IFFT
- 对IDFT公式两边取共轭

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

所以

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

先将X(k)取共轭，就可以直接利用FFT子程序，最后将结果再取一次共轭，并乘以1/N，即得到x(n)

作业：

一、1，3，4

二、实验：第十章的实验三。