

第4章

第4章 快速傅立叶变换 (FFT)

4.1 概述

4.2 时间抽取基 2 算法

4.3 频率抽取基 2 算法

4.4 减少运算量的措施

4.5 分裂基算法

4.6 线性调频 Z 变换

4.1 概述

各种识别、压缩算法等都涉及到快速计算要求

DFT的重要性：可以实现卷积、相关、滤波、谱估计等。DFT快速计算很重要。算法的快速计算都重要！

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi nk}{N}} \quad k = 0, 1, \dots, N-1$$
$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi nk}{N}} \quad n = 0, 1, \dots, N-1$$

若 $x(n)$ 是 N 点序列，实现 $x(n)$ 的DFT，即求出 N 个 $X(k)$ ，需要：

N^2 次复数乘法， $N(N-1)$ 次复数加法！

$$\begin{aligned}x(n): \quad N &= 1024, & N^2 &= 1048576 \\x(n_1, n_2): \quad N_1 &= N_2 = 512, & N^4 &= 262144^2 !\end{aligned}$$

若一次复数乘法需要 $100\mu\text{s}$ ，则要100s才能完成1024点的DFT

解决耗时的乘法问题是将数字信号处理理论用于实际的关键问题。特别是在早期，计算机的速度相当慢。因此，很多学者对解决DFT的快速计算问题产生了极大的兴趣。

Cooley J W, Tukey J W. An algorithm for the machine computation of complex Fourier series.

Mathematics of Computation, 1965, pp297~301

DSP的正式开端!

再举一个耗时的例子：用FFT算法处理一副 $N \times N$ 点的二维图像，若用每秒处理10万次复数乘法的计算机，当 $N = 1024$ 时，问需要多少时间（不考虑加法时间）？

解： $N = 1024$ ， $N^4 = 1099511627776 \approx 10^{12}$ （万亿量级），FFT算法处理二维图像，复数乘法约为

$$\frac{N^2}{2} \log_2 N^2 = 10485760 \approx 10^7 \text{（千万量级）}$$

而 $1024^4/10^7 \approx 1.0486e + 05$ （十万量级），即FFT算法仅为直接计算DFT所需时间的10万分之一。

直接计算DFT所需时间： $\frac{1024^4}{10\text{万} \cdot 3600} = 3054.2$ （小时），而FFT约需要时间： $\frac{10^7}{10\text{万} \cdot 60} = 1.6667$ （分钟）。

$$X(k) = \sum_{n=0}^{N-1} x(n)W^{nk} \quad k = 0, 1, \dots, N-1$$

FFT 的思路

1. $W = e^{-j2\pi/N}$ 的特殊值、周期性、对称性

(1) $W^0 = 1$

(2) $W^N = 1, W^{mN} = 1$

(3) $W^{N/2} = -1$

(4) $W^{N+r} = W^r$

(5) $W^{N/2+r} = -W^r$

W 的
特殊值

W 的周期性

W 的对称性

如何
充分
利用
这些
关系



FFT 的思路： 2, 3, 4

2. W 的可约性

$$W_N^{2rk} = W_{N/2}^{rk}$$

3. 长序列变成短序列

若 $N \rightarrow 2$ 个 $N/2$

则 N^2 次复述乘法 $\rightarrow 2(N/2)^2 = N^2/2$ 次复数乘法

4. 从信号的特殊性上考虑

如奇、偶、虚、实性

四点 DFT

$$X(k) = \sum_{n=0}^3 x(n)W^{nk} \quad k = 0,1,2,3$$

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$

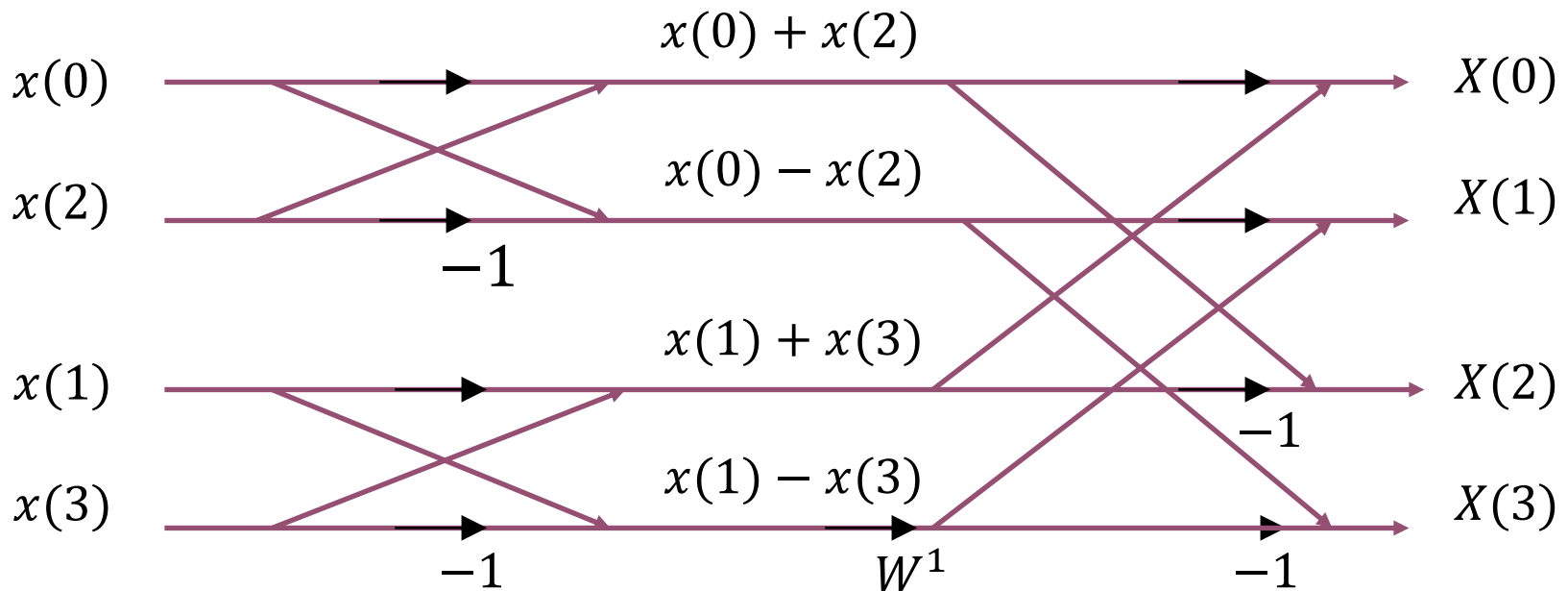
$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W^1 & -1 & -W^1 \\ 1 & -1 & 1 & -1 \\ 1 & -W^1 & -1 & W^1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \quad \text{交换}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & W^1 & -W^1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -W^1 & W^1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(2) \\ x(1) \\ x(3) \end{bmatrix}$$

$$\begin{aligned}
 X(0) &= [x(0) + x(2)] + [x(1) + x(3)] \\
 X(1) &= [x(0) - x(2)] + [x(1) - x(3)]W^1 \\
 X(2) &= [x(0) + x(2)] - [x(1) + x(3)] \\
 X(3) &= [x(0) - x(2)] - [x(1) - x(3)]W^1
 \end{aligned}$$

几个乘法？

$$W_4^1 = ?$$



FFT

库利-图基FFT算法

数字信号处理发展史的一个里程碑；DFT乘法计算量：

$N^2 \rightarrow \frac{N}{2} \log_2^N, N = 1024$ ，计算量降为5120次，仅为原来的0.48828125%

FFT算法分类

N 为2的整数次幂：基2、基4、分裂基算法

N 不为2的整数次幂：Winograd算法，素因子算法。建立在下标影射 N 的乘法次数明显减少；理论复杂、编程困难；内存开销、数据传递量增多；在新的DSP技术下，优势不再明显

DSP技术发展

一个指令周期完成一次乘累加（乘法和加法时间相当）

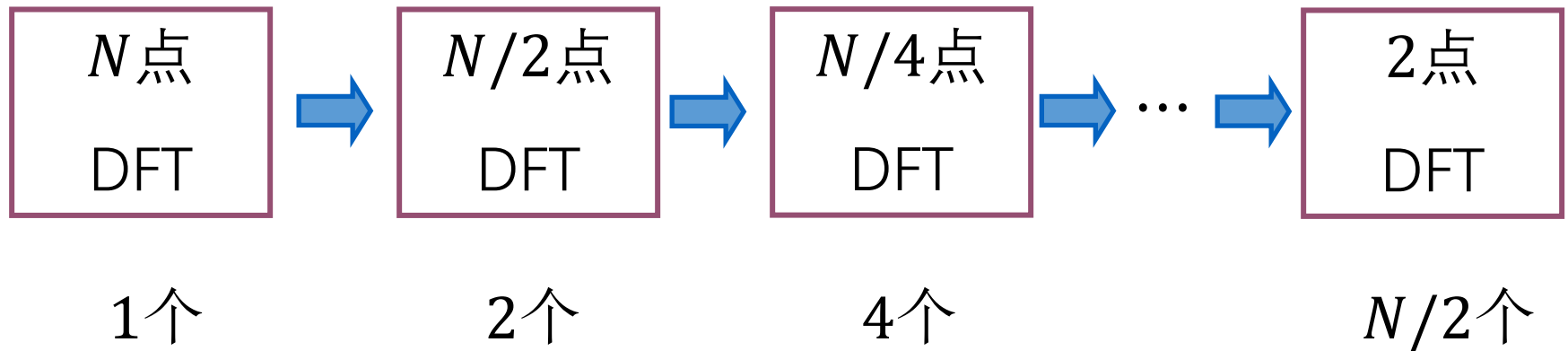
数据传递时间需要考虑（相对于运算时间不能忽略）

存算一体技术/芯片

存算一体技术概念的形成，最早可以追溯到上个世纪90年代。随着近几年云计算和人工智能（AI）应用的发展，面对计算中心的数据洪流，数据搬运慢、搬运能耗大等问题成为了计算的关键瓶颈。从处理单元外的存储器提取数据，搬运时间往往是运算时间的成百上千倍，整个过程的无用能耗大概在60%-90%之间，能效非常低，**“存储墙”成为了数据计算应用的一大障碍**。深度学习加速的最大挑战就是数据在计算单元和存储单元之间频繁的移动。……

4.2 时间抽取基 2 算法

FFT的核心思想是：



问题：如何分解最有效？

可以对时间变量分解 (DIT)，也可对频率变量分解(DIF)

对时间序列 $x(n)$ 按自变量 n 奇偶分成两组

令：

$$\left. \begin{array}{l} n = 2r \\ n = 2r + 1 \end{array} \right\} r = 0, 1, \dots, N/2 - 1$$

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} \\ &= \sum_{r=0}^{N/2-1} x(2r) W_N^{2rk} + \sum_{r=0}^{N/2-1} x(2r+1) W_N^{(2r+1)k} \\ &= \sum_{r=0}^{N/2-1} x(2r) W_{N/2}^{rk} + W_N^k \sum_{r=0}^{N/2-1} x(2r+1) W_{N/2}^{rk} \end{aligned}$$

可约性

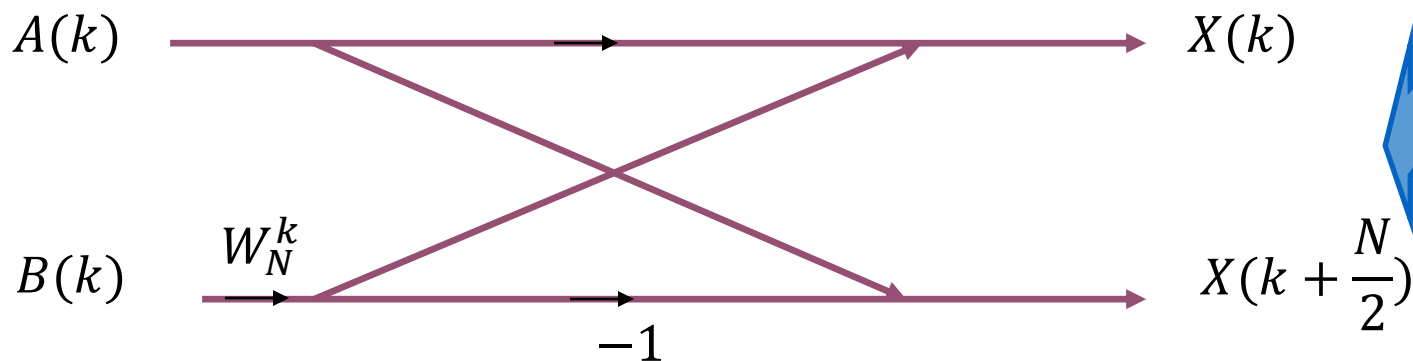
$$X(k) = \underbrace{\sum_{r=0}^{N/2-1} x(2r)W_{N/2}^{rk}}_{A(k)} + W_N^k \underbrace{\sum_{r=0}^{N/2-1} x(2r+1)W_{N/2}^{rk}}_{B(k)}$$

$k = 0, 1, \dots, N/2 - 1$

$$X(k) = A(k) + W_N^k B(k) \quad k = 0, 1, \dots, \frac{N}{2} - 1$$

$$X(k + \frac{N}{2}) = A(k) - W_N^k B(k) \quad k = 0, 1, \dots, \frac{N}{2} - 1$$

对称性: $W_N^{k+N/2} = W_N^k W_N^{N/2} = -W_N^k$



$A(k)$ 、 $B(k)$ 都是 $N/2$ 点的 DFT，它们各自又可分成 $N/4$ 点的DFT。 $A(k)$ 、 $B(k)$ 分别朝短序列分解。

$$2r \xrightarrow{r=2l, r=2l+1} \begin{cases} 4l \\ 4l+2 \end{cases}$$

推导并化简 $A(k)$ 的计算，
并用 $C(k)$ 和 $D(k)$ 表示

主要利用了

可约性 $W_{N/2}^{2lk} = W_{N/4}^{lk}$

对称性 $W_{N/2}^{N/4} = -1$

$$C(k) = \sum_{l=0}^{N/4-1} x(4l) W_{N/4}^{lk}, k = 0, 1, \dots, N/4 - 1$$

$$D(k) = \sum_{l=0}^{N/4-1} x(4l+2) W_{N/4}^{lk}, k = 0, 1, \dots, N/4 - 1$$

$$A(k) = C(k) + W_{N/2}^k D(k), k = 0, 1, \dots, N/4 - 1$$

$$A(k + N/4) = C(k) - W_{N/2}^k D(k), k = 0, 1, \dots, N/4 - 1$$

$$2r + 1 \xrightarrow{r=2l, r=2l+1} \begin{cases} 4l + 1 \\ 4l + 3 \end{cases}$$

推导并化简 $B(k)$ 的计算,
并用 $E(k)$ 和 $F(k)$ 表示

$$E(k) = \sum_{l=0}^{N/4-1} x(4l + 1) W_{N/4}^{lk}, k = 0, 1, \dots, N/4 - 1$$

$$F(k) = \sum_{l=0}^{N/4-1} x(4l + 3) W_{N/4}^{lk}, k = 0, 1, \dots, N/4 - 1$$

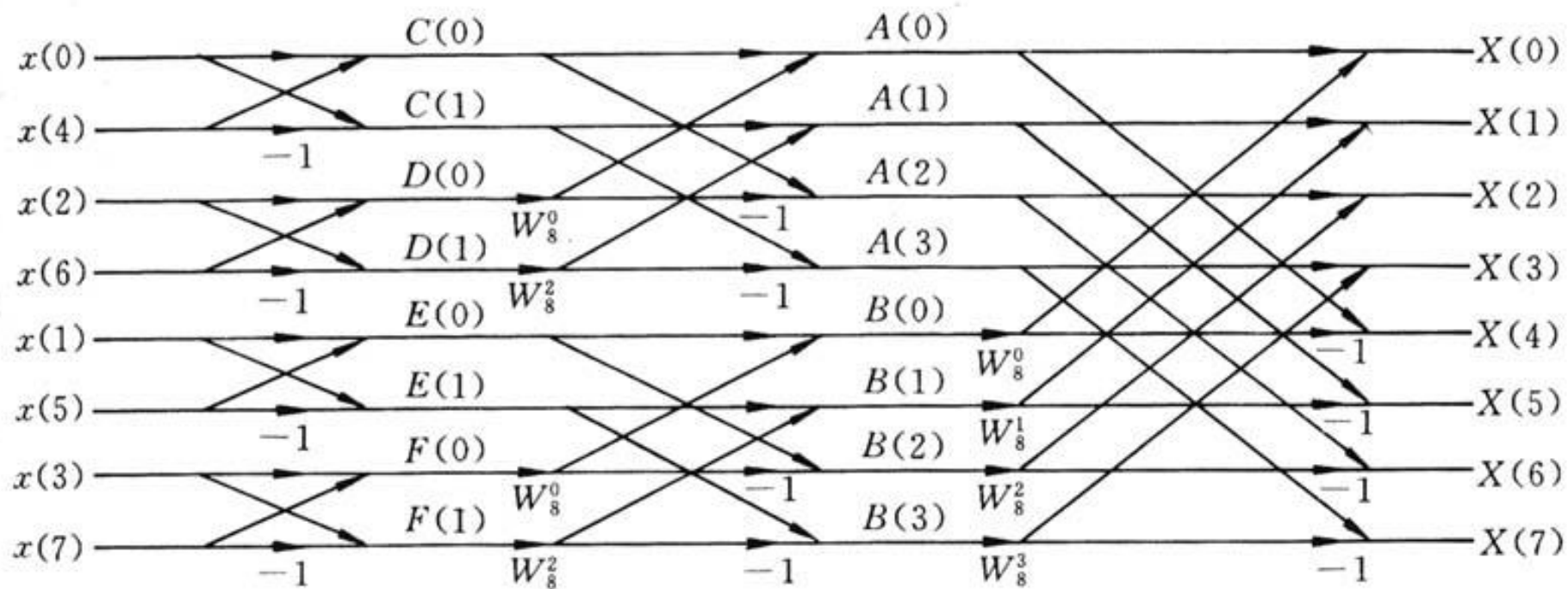
$$B(k) = E(k) + W_{N/2}^k F(k), k = 0, 1, \dots, N/4 - 1$$

$$B(k + N/4) = E(k) - W_{N/2}^k F(k), k = 0, 1, \dots, N/4 - 1$$

如此继续分下去，直至两点DFT。两点DFT不需要乘法运算：

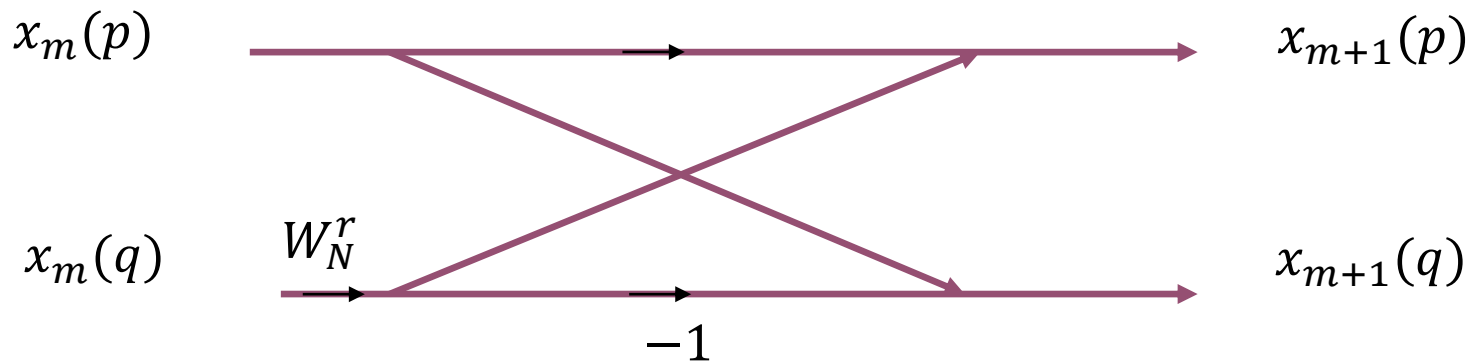
$$\begin{aligned} X(0) &= x(0) + x(1) \\ X(1) &= x(0) - x(1) \end{aligned}$$

对 $N = 2^M$ ，共可分 M 次，即 $m = 0, 1, \dots, M - 1$



8点FFT时间抽取算法信号流图

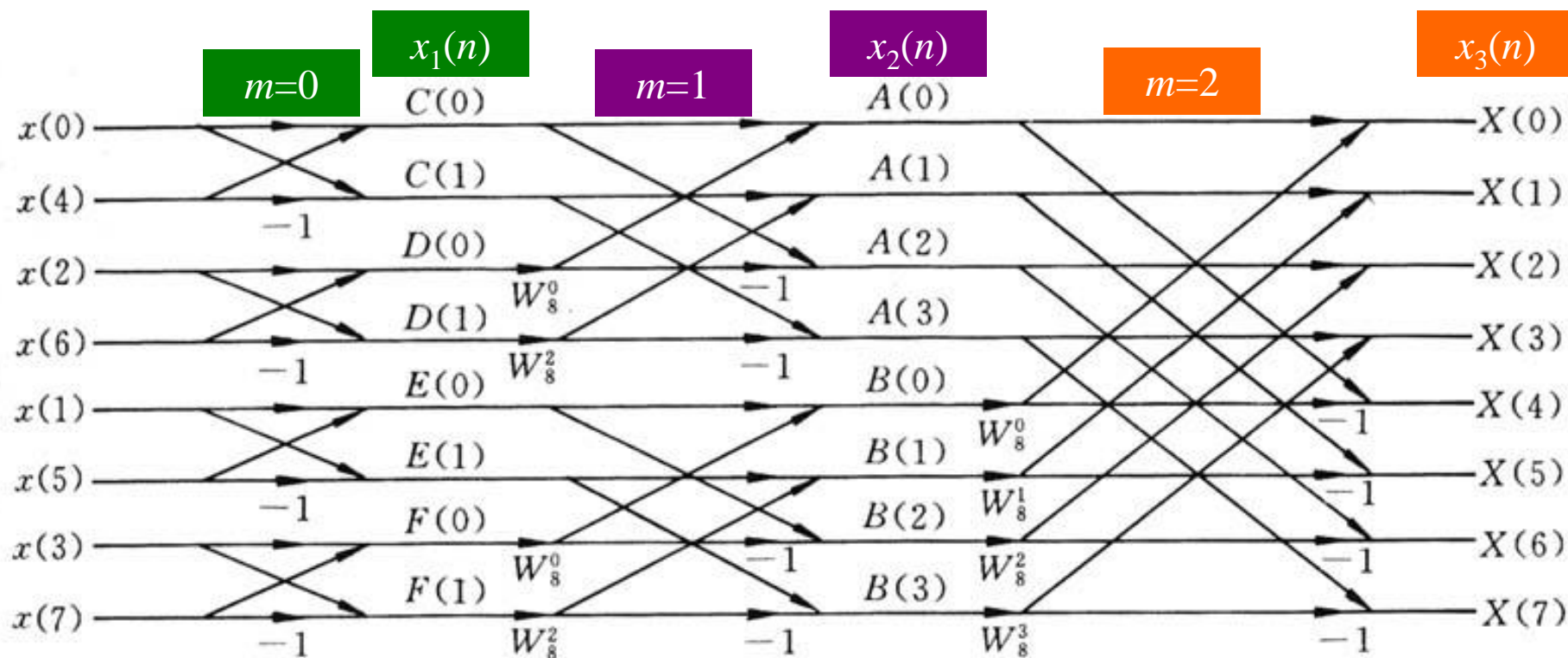
每一级有 $N/2$ 个如下的“蝶形”单元：



算法讨论：“级”的概念、碟形单元、“组”的概念、旋转因子的分布、码位倒置

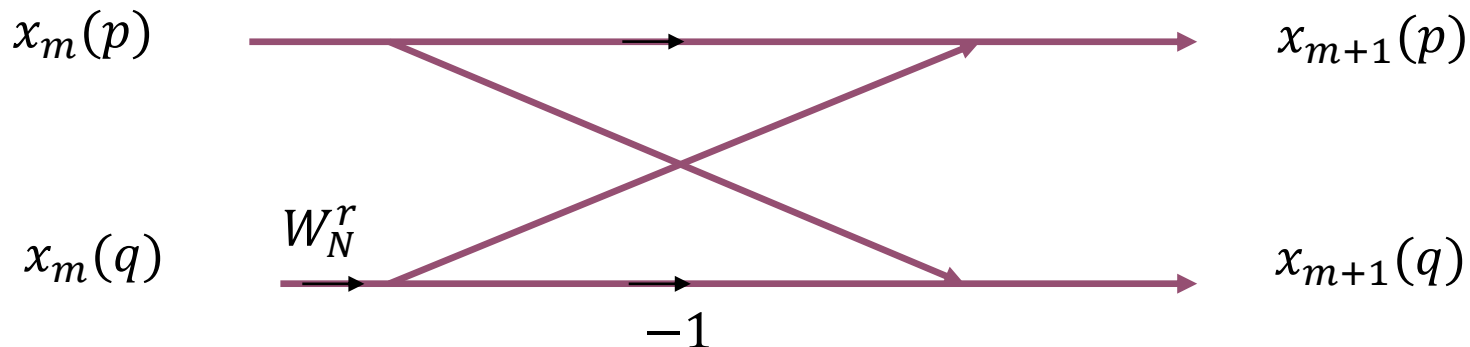
“级”的概念

- 分解级数 $M = \log_2 N$
- 记录级数的变量为 m , $m = 0, 1, \dots, M - 1$



8点FFT时间抽取算法信号流图

蝶形单元

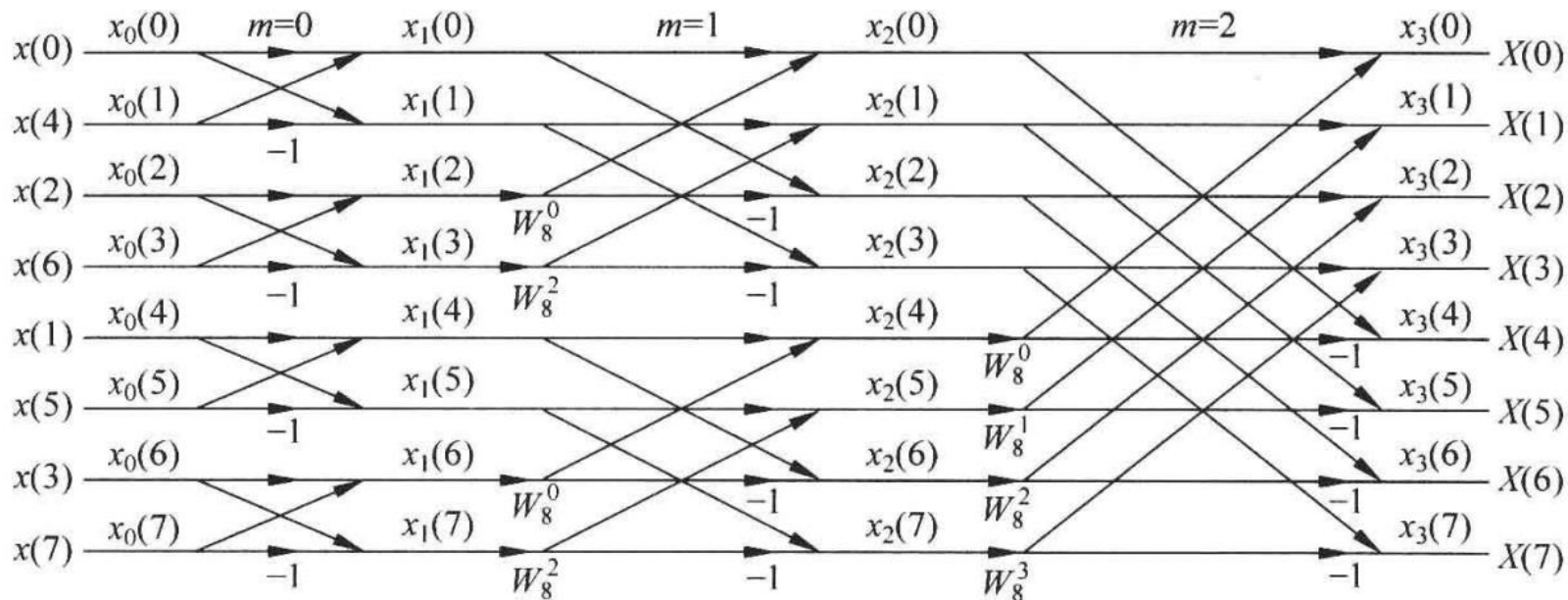


$$x_{m+1}(p) = x_m(p) + x_m(q)W_N^r$$

$$x_{m+1}(q) = x_m(p) - x_m(q)W_N^r$$

先乘后
再加减

即：每一个蝶形单元仅需一个复数乘法，两个复数加法。两点构成一个蝶形单元，并且这两点不再参与别的蝶形单元的运算。同址运算。



变量 p 、 q 是参与碟形单元运算的上、下节点的序号。在第 m 级（每级的数据自上而下按自然顺序排列），上下节点 p 、 q 之间的距离为 $q-p=2^m$ 。

节点距离用 B 表示： $B=q-p=2^m$ 。

$m=0$ 级， $B=1$ ：1-0，3-2，……；

$m=1$ 级， $B=2$ ：2-0，3-1，……；

$m=2$ 级， $B=4$ ：4-0，5-1，……；

运算量分析：每一级有 $N/2$ 个碟形单元，每个碟形单元只需一次复数乘法、两次复数加法。对于 M 级，复数乘法次数 M_c 、复数加法次数 M_a ：

$$M_c = \frac{N}{2} M = \frac{N}{2} \log_2 N$$

$$M_a = NM = N \log_2 N$$

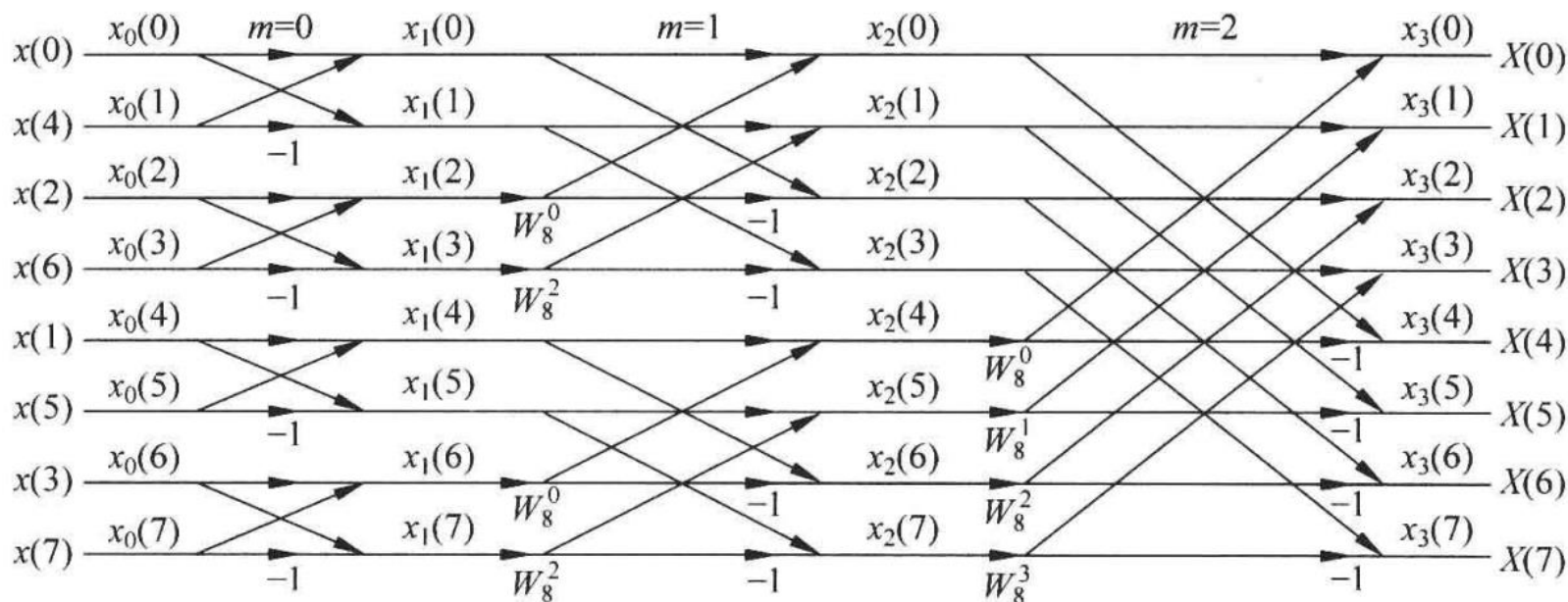
“组”的概念

每一级的 $N/2$ 个碟形单元划分成若干组，每组有相同的结构和旋转因子 W^r 的分布

$m = 0$ 级，有四组； $m = 1$ 级，有两组； $m = 2$ 级，有一组

m 级每组数据点数 N_m ： 2^{m+1}

m 级组数变量 G_m ： $\frac{N}{2^{m+1}} = 2^{M-1-m}$



旋转因子 W^r 的位置与分布

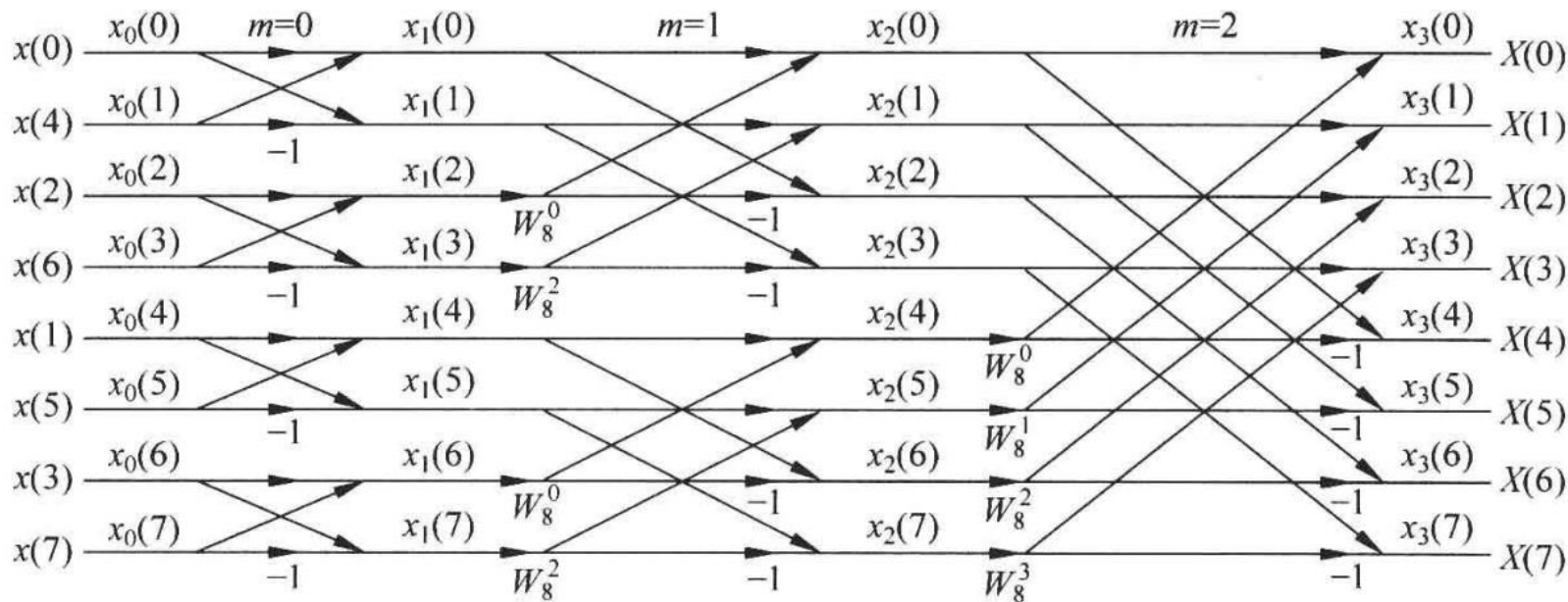
考虑级数变量 m 和每级数据点数，每级的旋转因子 $W_{2^{m+1}}^r$ 为：

$$W_{2^{m+1}}^r, r = 0, 1, \dots, (2^m - 1), m = 0, 1, \dots, (\log_2^N - 1)$$

第 m 级每组点数为： $N_m = 2^{m+1}$

r 取值有： $2^m (= N_m/2)$ 个

$$W_{2^{m+1}}^r = W_{N_m}^r$$



DSP有专门的码位
倒置寻址指令

码位倒置 (输出有序, 输入倒序)

n	$x(n)$	$X(k)$	k
0	000	000	0
4	100	001	1
2	010	010	2
6	110	011	3
1	001	100	4
5	101	101	5
3	011	110	6
7	111	111	7




```
function y=MATLAB_ditfft(x)
    m=nextpow2(length(x));
    N=2^m; % 求x的长度对应的2的最低幂次m
    if length(x)<N
        x=[x,zeros(1,N-length(x))]; % 若x的长度不是2的幂，补零到2的整数幂
    end
    %nxd=bin2dec(fliplr(dec2bin([1:N]-1,m)))+1; % 求1:2^m数列的倒序
    nxd0 = dec2bin([1:N]-1,m);
    nxd1 = fliplr(nxd0);
    nxd=bin2dec(nxd1)+1;
    y=x(nxd); % 将x倒序排列作为y的初始值
    for mm=1:m % 将DFT作m次基2分解,从左到右，对每次分解作DFT运算
        Nmr=2^mm;u=1; % 旋转因子u初始化为WN^0=1
        WN=exp(-1i*2*pi/Nmr); % 本次分解的基本DFT因子WN=exp(-i*2*pi/Nmr)
        for j=1:Nmr/2 % 本次跨越间隔内的各次蝶形运算
            for k=j:Nmr:N % 本次蝶形运算的跨越间隔为Nmr=2^mm
                kp=k+Nmr/2; % 确定蝶形运算的对应单元下标
                t=y(kp)*u; % 蝶形运算的乘积项
                y(kp)=y(k)-t; % 蝶形运算
                y(k)=y(k)+t; % 蝶形运算
            end
            u=u*WN; % 修改旋转因子,多乘一个基本DFT因子WN
        end
    end
end
end
```

4.3 频率抽取基 2 算法

$$\begin{aligned} X(k) &= \sum_{n=0}^{N/2-1} x(n)W_N^{nk} + \sum_{n=N/2}^{N-1} x(n)W_N^{nk} \\ &= \sum_{n=0}^{N/2-1} x(n)W_N^{nk} + \sum_{n=0}^{N/2-1} x(n + N/2)W_N^{nk}W_N^{Nk/2} \\ &= \sum_{n=0}^{N/2-1} [x(n) + (-1)^{-k}x(n + N/2)]W_N^{nk} \end{aligned}$$

时域序列
前后分半

不对应正
常的公式

$$\begin{aligned} \text{令: } & k = 2r, \quad k = 2r + 1 \\ & r = 0, 1, \dots, N/2 - 1 \end{aligned}$$

频域按奇
偶性抽取

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

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

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

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

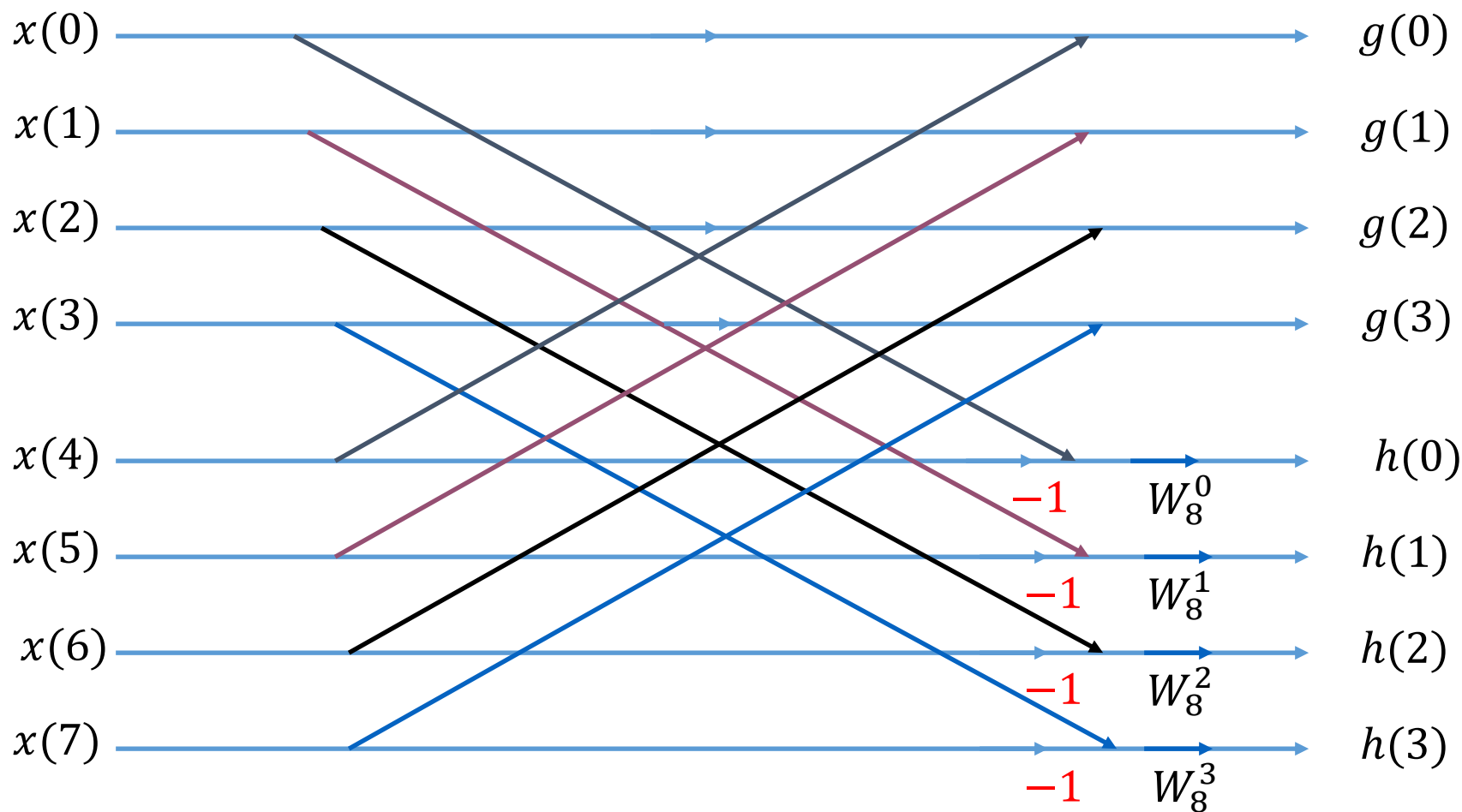
各是 $N/2$ 点的DFT

$$g(n) = x(n) + x(n + N/2)$$


$$h(n) = [x(n) - x(n + N/2)]W_N^n$$

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

由此得到基本运算单元



$$\begin{aligned}
 X(2r) &= \sum_{n=0}^{N/2-1} g(n) W_{N/2}^{nr} \\
 X(2r+1) &= \sum_{n=0}^{N/2-1} h(n) W_{N/2}^{nr}
 \end{aligned}$$



 各是 $N/2$ 点的
DFT

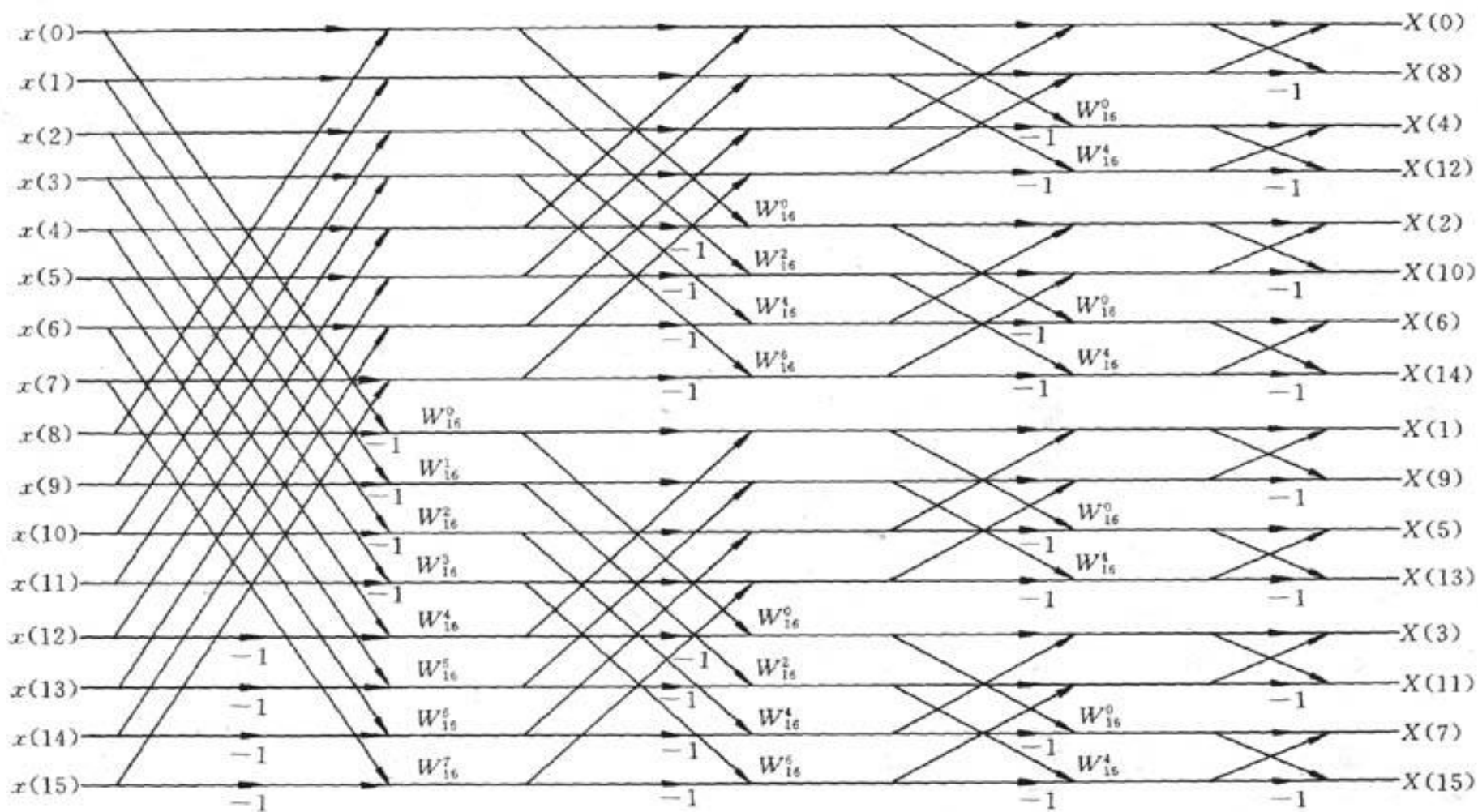
继续分解，直到两点DFT

将 n 分解：Decimation In Time, DIT 时间抽取

将 k 分解：Decimation In Freq., DIF 频率抽取

对应着DFT变换矩阵 W 的不同分解

注意 DIT 和 DIF 的对偶性质



输入正序，输出倒序。注意 W^r 因子的位置

4.4 进一步减少运算量的措施

FFT中乘法运算主要来自和复指数相乘

$$W_N^r, \quad r = 0, 1, \dots, N/2 - 1 \quad (1 \text{ 组})$$

$$W_{N/2}^r, \quad r = 0, 1, \dots, N/4 - 1 \quad (2 \text{ 组})$$

$$W_{N/4}^r, \quad r = 0, 1, \dots, N/8 - 1 \quad (4 \text{ 组})$$

\vdots

$$W_4^r, \quad r = 0, 1 \quad (N/4 \text{ 组})$$

$$W_2^r, \quad r = 0 \quad (N/2 \text{ 组})$$

复数乘法次数

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

旋转因子

twiddle factor

$$m = 0 \quad W_2^r, \quad r = 0$$

$$m = 1 \quad W_4^r, \quad r = 0,1$$

$$m = 2 \quad W_8^r, \quad r = 0,1,2,3$$

\vdots

$$m = M - 1 \quad W_N^r, \quad r = 0,1,\dots,\frac{N}{2} - 1$$

$$\left. \begin{array}{l} W^0 = 1 \\ W_4^1 = -j \end{array} \right\}$$

不需要乘法，无关紧要的旋转因子 (trivial ~)

M 级，前两级都是 W^0 , W_4^1 ，去除之，则：

$$M_c = \frac{N}{2}(M - 2)$$

后 $M - 2$ 级，含有

$$\frac{N}{4} + \frac{N}{8} + \cdots + \frac{N}{N/2} = \frac{N}{2} - 2 \uparrow W_N^0, W_4^1$$

例如， $m = 2$ 级：每组有 W_8^0 、 W_8^2 ，共有 $N/2^{2+1} = N/8$ 组，
则无关紧要旋转因子个数为 $N/4$ 个；以此类推，……

再去除之，则有：

$$M_c = \frac{N}{2}(M - 3) + 2 \text{ (复乘)}$$

$$W_8^1 = (1 - j)\sqrt{2}/2 = c - jc \quad \Rightarrow \quad \text{虚部和实部相等, trivial ~}$$

两个复数相乘，需要四次实乘、两次实加。实现和 W_8^1 的相乘，需两次实乘，两次实加。

N 点FFT中，有多少个类似 W_8^1 的操作？

$$\frac{N}{4} + \frac{N}{8} + \cdots + \frac{N}{N/2} = \frac{N}{2} - 2 \quad \uparrow W_8^1$$

将所有无关紧要旋转因子去除，或单独考虑，有：

$$M_R = 4 \left[\frac{N}{2} (M - 3) + 2 - \left(\frac{N}{2} - 2 \right) \right] + 2 \left(\frac{N}{2} - 2 \right)$$

$$\begin{cases} M_R = N(2M - 7) + 12 & \text{实乘} \\ A_R = 3N(M - 1) + 4 & \text{实加} \end{cases}$$



各种算法比较的基础

- 措施
1. 以上称为多蝶形单元运算;
 2. 单独处理实数据的输入;
 3. 采用新的 FFT 算法。

一个 N 点 FFT 程序同时计算两个 N 点实序列的 DFT
 一个 N 点实序列的 DFT 用 $N/2$ 点的 FFT 程序实现

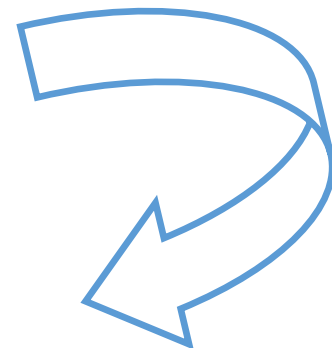


表 5.4.1 基 2 FFT 在各种蝶形单元下所需的实数乘法和实数加法(M_1, A_1, i
对应蝶形单元数,一个普通复数乘法使用四次实数乘和二次实数加)

蝶形单元 N	一类蝶形单元		二类蝶形单元		三类蝶形单元		四类蝶形单元	
	M_1	A_1	M_2	A_2	M_3	A_3	M_4	A_4
2	4	6	0	4	0	4	0	4
8	48	72	20	58	8	52	4	52
32	320	480	196	418	136	388	108	288
128	1 792	2 688	1 284	2 434	1 032	2 308	908	2 308
512	9 216	13 824	7 172	12 802	6 152	12 292	5 644	12 292
2 048	45 056	67 584	36 868	63 490	32 776	61 444	30 732	61 444
4 096	98 304	14 756	81 924	139 266	73 736	135 172	69 644	135 172

包含所有
旋转因子

去掉 ± 1

去掉 $\pm j$

特殊处理

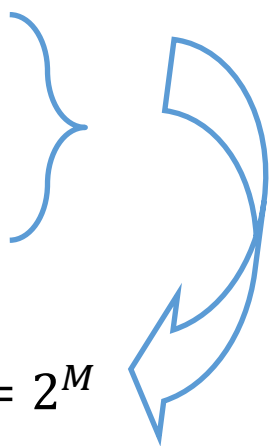
多蝶形单元运算所需计算量的比较

4.5 分裂基 (Split-radix) 算法

基-2 算法： 1965年， DSP 发展的里程碑；

基-4 算法： 对基-2 算法的改进；

分裂基算法： 1984年， 接近最优的 FFT!


$$N = 2^M$$

Winograd 算法： 1976年提出， 是具有鲜明特色的FFT! 用到较多的数论知识， 可用于 N 不等于2的整次幂。

基4 DIF FFT算法

基本思路

时间序列长度满足4的整次幂 $N = 4^M$

对时间序列按时间自变量顺序分成4段

针对4个子部分的求和范围和旋转因子进行处理以符合DFT的计算公式：求和范围从0开始，到 $N/4-1$ ；旋转因子的下标指示变换序列的长度 ($N/4$)

基本方法

对频域序列变量 k 进行抽取。令 $k=4r, 4r+2, 4r+1, 4r+3, r=0,1,\dots, N/4-1$

详细推导 $X(r)$ 、 $X(r+2)$ ，利用旋转因子的性质化简计算

基本特点

对于 $N=16$ ，分解为2两级， $m=0, 1$ ；掌握基本单元信号流图

$$X(k) = \sum_{n=0}^{N/4-1} x(n)W_N^{kn} + \sum_{n=N/4}^{N/2-1} x(n)W_N^{kn} + \sum_{n=N/2}^{3N/4-1} x(n)W_N^{kn} + \sum_{n=3N/4}^{N-1} x(n)W_N^{kn}$$

$$\text{令 } k=4r, \quad 4r+2, \quad 4r+1, \quad 4r+3, \quad r=0,1,\dots,N/4-1$$

$$\begin{aligned} X(4r) &= \sum_{n=0}^{N/4-1} x(n)W_N^{4rn} + \sum_{n=0}^{N/4-1} x(n + \frac{N}{4})W_N^{4r(n+N/4)} + \\ &\quad \sum_{n=0}^{N/4-1} x(n + \frac{N}{2})W_N^{4r(n+N/2)} + \sum_{n=0}^{N/4-1} x(n + \frac{3N}{4})W_N^{4r(n+3N/4)} \\ &= \sum_{n=0}^{N/4-1} x(n)W_{N/4}^{rn} + \sum_{n=0}^{N/4-1} x(n + \frac{N}{4})W_N^{rN}W_{N/4}^{rn} + \\ &\quad \sum_{n=0}^{N/4-1} x(n + \frac{N}{2})W_N^{2rN}W_{N/4}^{rn} + \sum_{n=0}^{N/4-1} x(n + \frac{3N}{4})W_N^{3rN}W_{N/4}^{rn} \end{aligned}$$

$$\begin{aligned} W_N^{rN} &= 1 \\ W_N^{2rN} &= 1 \\ W_N^{3rN} &= 1 \end{aligned}$$

$$X(4r) = \sum_{n=0}^{N/4-1} \left[\left(x(n) + x(n + \frac{N}{2}) \right) + \left(x(n + \frac{N}{4}) + x(n + \frac{3N}{4}) \right) \right] W_{N/4}^{rn}$$

$$\begin{aligned}
X(4r + 2) &= \sum_{n=0}^{N/4-1} x(n) W_N^{(4r+2)n} + \sum_{n=0}^{N/4-1} x\left(n + \frac{N}{4}\right) W_N^{(4r+2)(n+N/4)} + \\
&\quad \sum_{n=0}^{N/4-1} x\left(n + \frac{N}{2}\right) W_N^{(4r+2)(n+N/2)} + \sum_{n=0}^{N/4-1} x\left(n + \frac{3N}{4}\right) W_N^{(4r+2)(n+3N/4)} \\
&= \sum_{n=0}^{N/4-1} x(n) W_N^{2n} W_{N/4}^{rn} + \sum_{n=0}^{N/4-1} x\left(n + \frac{N}{4}\right) W_N^{2n} W_N^{N/2} W_N^{rN} W_{N/4}^{rn} + \\
&\quad \sum_{n=0}^{N/4-1} x\left(n + \frac{N}{2}\right) W_N^{2n} W_N^N W_N^{2rN} W_{N/4}^{rn} + \sum_{n=0}^{N/4-1} x\left(n + \frac{3N}{4}\right) W_N^{2n} W_N^{3N/2} W_N^{3rN} W_{N/4}^{rn}
\end{aligned}$$

$$W_N^{rN} = W_N^{2rN} = W_N^{3rN} = 1, \quad W_N^{N/2} = W_N^{3N/2} = -1$$

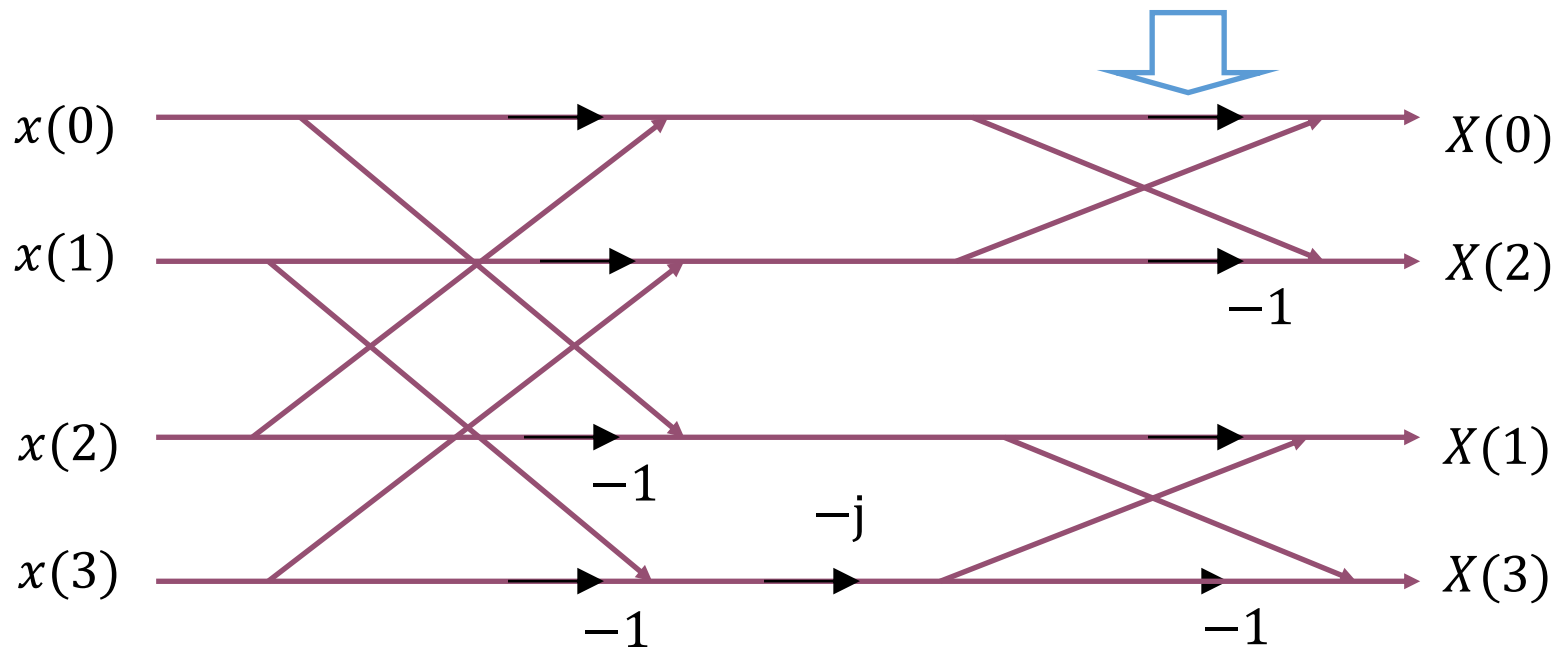
$$X(4r + 2) = \sum_{n=0}^{N/4-1} \left[\left(x(n) + x\left(n + \frac{N}{2}\right) \right) - \left(x\left(n + \frac{N}{4}\right) + x\left(n + \frac{3N}{4}\right) \right) \right] W_N^{2n} W_{N/4}^{rn}$$

$$X(4r + 1) = \sum_{n=0}^{N/4-1} \left[\left(x(n) - x\left(n + \frac{N}{2}\right) \right) - j \left(x\left(n + \frac{N}{4}\right) - x\left(n + \frac{3N}{4}\right) \right) \right] W_N^n W_{N/4}^{rn}$$

$$X(4r + 3) = \sum_{n=0}^{N/4-1} \left[\left(x(n) - x\left(n + \frac{N}{2}\right) \right) + j \left(x\left(n + \frac{N}{4}\right) - x\left(n + \frac{3N}{4}\right) \right) \right] W_N^{3n} W_{N/4}^{rn}$$

基4 DIF 的基本单元

不需要乘法!

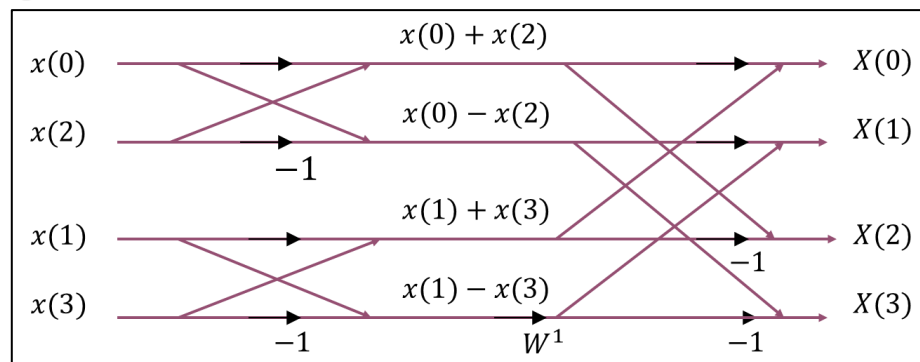


以 4 为基，分解时级数可减少一半，因此可减少乘法次数。

$$N = 4^M = 2^{2M}$$

乘法数减少一半?

基 4 DIT
基本单元



分裂基算法推导

基2 和基4 算法的比较:

基2 DIF

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

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

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

可见，**旋转因子**都出现在奇序号项输出，在求出偶序号项时不需要乘法。每一级都是如此。

基4 DIF

$$\text{令} \begin{cases} a(n) = x(n) + x(n + \frac{N}{2}) & c(n) = x(n + \frac{N}{4}) + x(n + \frac{3N}{4}) \\ b(n) = x(n) - x(n + \frac{N}{2}) & d(n) = x(n + \frac{N}{4}) - x(n + \frac{3N}{4}) \end{cases}$$

则

$$X(4r) = \sum_{n=0}^{N/4-1} [a(n) + c(n)] W_{N/4}^{nr}$$

$$X(4r + 2) = \sum_{n=0}^{N/4-1} [a(n) - c(n)] \underline{W_N^{2n}} W_{N/4}^{nr}$$

$$X(4r + 1) = \sum_{n=0}^{N/4-1} [b(n) - jd(n)] \underline{W_N^n} W_{N/4}^{nr}$$

$$X(4r + 3) = \sum_{n=0}^{N/4-1} [b(n) + jd(n)] \underline{W_N^{3n}} W_{N/4}^{nr}$$

分析上述结果可知：在基-4算法中， $N/4$ 个偶序号输出也要乘 W 因子。而基-2算法的偶序号项都不要乘 W 因子。

请思考：

如何将基-2和基-4的优点都兼收？

对偶序号项输出用基-2算法，对奇序号项输出用基-4算法。

⇒ 分裂基算法

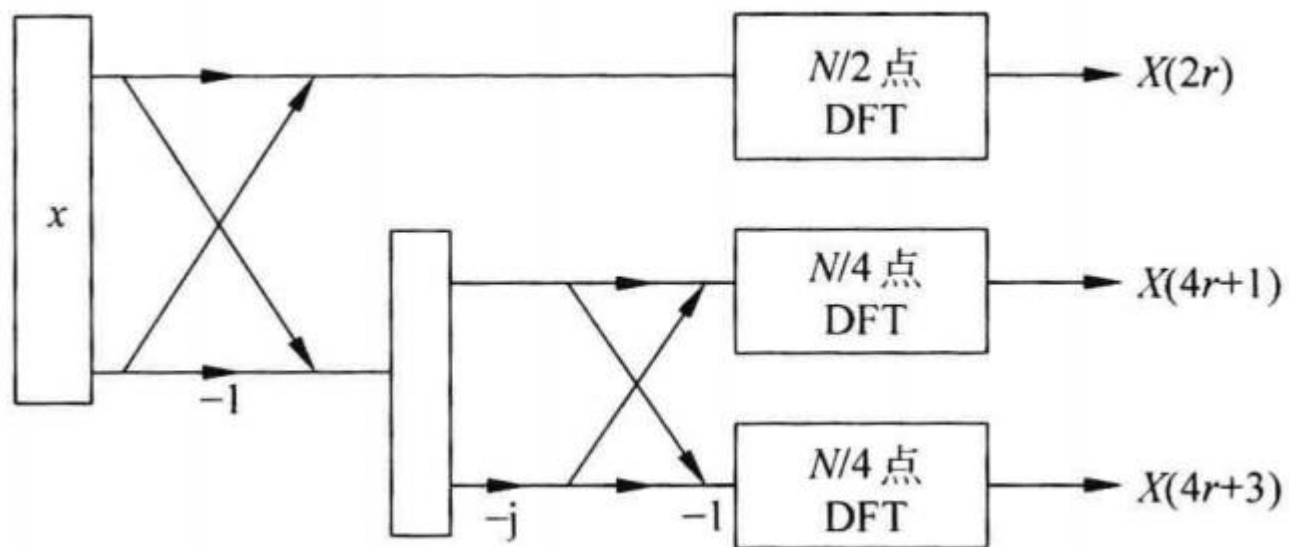
$$\text{令} \begin{cases} a(n) = x(n) + x(n + \frac{N}{2}) & \blacksquare(n) = x(n + \frac{N}{4}) + x(n + \frac{3N}{4}) \\ b(n) = x(n) - x(n + \frac{N}{2}) & c(n) = x(n + \frac{N}{4}) - x(n + \frac{3N}{4}) \end{cases}$$

奇偶分开处理：偶： $2r$ ，即 $4r$ 和 $4r+2$ ；奇： $4r+1$ 和 $4r+3$

$$\text{则} \begin{cases} X(2r) = \sum_{n=0}^{N/2-1} a(n) W_{N/2}^{nr} \\ X(4r+1) = \sum_{n=0}^{N/4-1} \underbrace{[b(n) - jc(n)] W_N^n W_{N/4}^{nr}}_{d(n)} \\ X(4r+3) = \sum_{n=0}^{N/4-1} \underbrace{[b(n) + jc(n)] W_N^{3n} W_{N/4}^{nr}}_{e(n)} \end{cases}$$



基 2 / 4 算法



分裂基算法的示意图

对未达到4点的短序列再继续分解

例如，设 $X(2r)$, $r=0,1,\dots,7$ ，对应时间序列为 $a(n)$

先对时间序列 $a(n)$ 按时间自变量顺序分成两部分，每部分对应应有4点长度。

- {由于 $a(n)$ 已经是从小 $x(n)$ 分出来的，所以这里更短的4点长度的子序列实质上可以用原始序列 $x(n)$ 各部分组合表示。}

再令 $r=2l$, $4l+1$, $4l+3$ ，即对频域变量 r 进行抽取（同第1级）

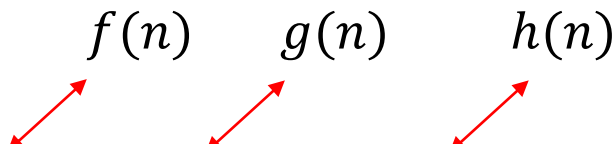
$X(0), X(2), X(4), X(6), X(8), X(10), X(12), X(14)$

$r=2l$: $X(0), X(4), X(8), X(12)$, $l=0,1,2,3$

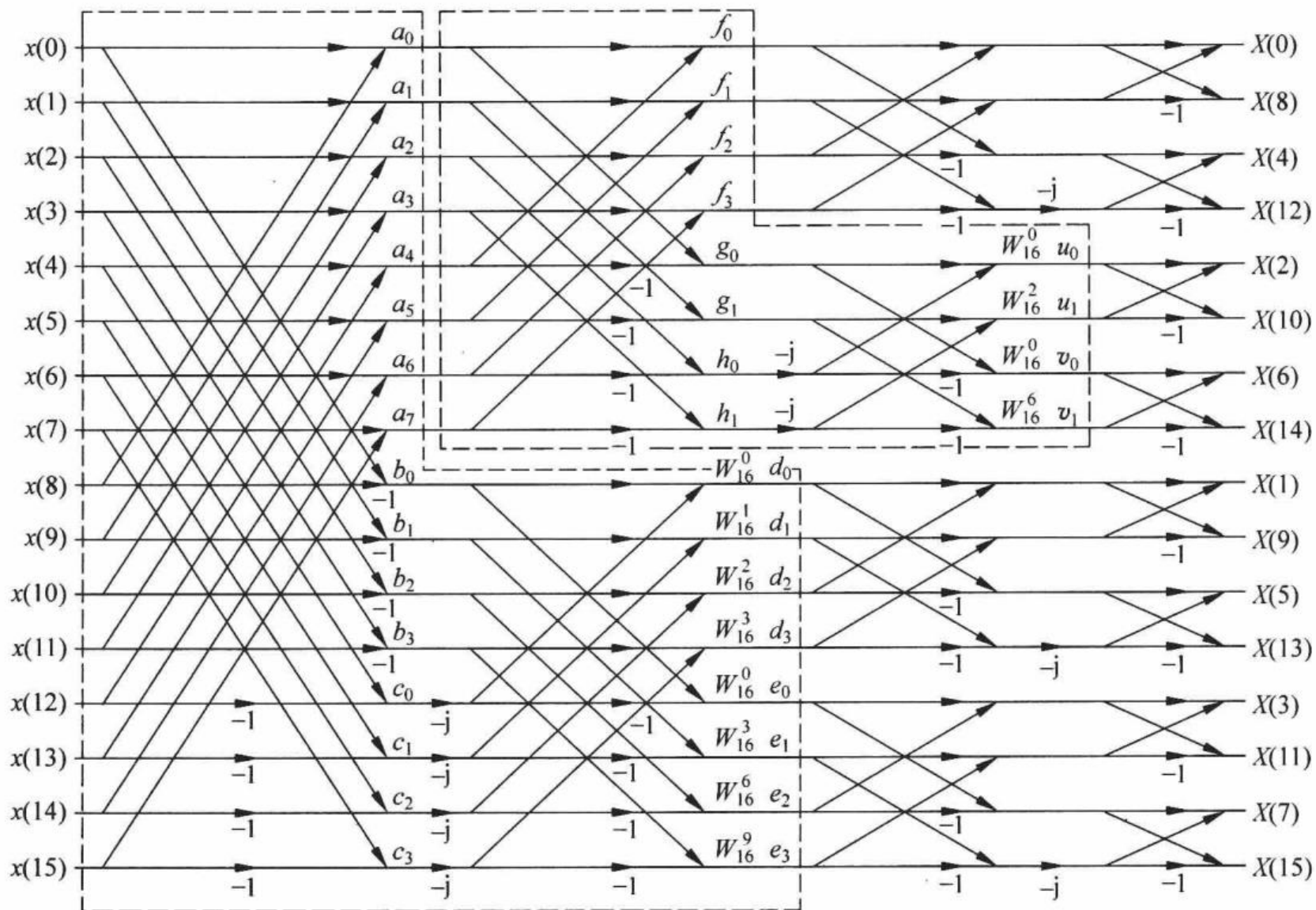
$4l+1$: $X(2), X(10)$, $l=0,1$

$4l+3$: $X(6), X(14)$, $l=0,1$

最后分别计算3个序列： $X(4l)$, $X(8l+2)$, $X(8l+6)$



$f(n)$ $g(n)$ $h(n)$



16点分裂基算法信号流图

所需计算量

基2

$$\begin{cases} M_R = 2MN - 7N + 12 \\ A_R = 3MN - 3N + 4 \end{cases}$$

基4

$$\begin{cases} M_R = \frac{3}{2}MN - 5N + 8 \\ A_R = \frac{11}{4}MN - \frac{13}{6}N + \frac{8}{3} \end{cases}$$

分裂基

$$\begin{cases} M_R = \frac{4}{3}MN - \frac{38}{9}N + 6 + (-1)^M \frac{2}{9} \\ A_R = \frac{8}{3}MN - \frac{16}{9}N + 2 - (-1)^M \frac{2}{9} \end{cases}$$

注意导出方法

分裂基算法
计算量推导

极限： $M_R = 2N - M^2 - M - 2$

2. 分裂基算法的计算量

分析(4.5.3)式可以看出,一个 N 点 DFT 在第一级被分成了一个 $N/2$ 点 DFT 和两个 $N/4$ 点的 DFT。 $N/2$ 点 DFT 对应偶序号输出,不包含 W 因子。两个 $N/4$ 点 DFT 对应奇序号输出,共有 $N/2$ 个旋转因子,其中包含两个 W^0 因子,两个 W_8^1 因子,它们都可以特殊处理,因此,这一级应需要 $(N/2 - 4)$ 个一般复数乘和两个乘以 W_8^1 的特殊复数乘。若实现一次复数乘需四次实数乘、两次实数加,那么实现这一级运算共需要 $[4(N/2 - 4) + 2 \times 2 = 2N - 12]$ 次实数乘法。由此可得到递推公式

$$Q_n = Q_{n-1} + 2Q_{n-2} + 2 \times 2^n - 12 \quad (4.5.5)$$

式中 $n = 3, 4, \dots, M$, 而 $M = \log_2 N$, Q_n 代表 $N = 2^n$ 时所需要的乘法量,初始条件是 $Q_1 = 0, Q_2 = 0$ 。现在来求解这一差分方程。不妨将(4.5.5)式写成

$$x(n) = x(n-1) + 2x(n-2) + 2^{n+1} - 12 \quad (4.5.6)$$

假定 N 为 ∞ , 则 M 也为无穷,此时对上式两边取 Z 变换(注意 n 的值从 3 开始),得

$$\begin{aligned} & X(z)[1 - z^{-1} - 2z^{-2}] \\ &= 2 \left[\frac{1}{1 - 2z^{-1}} - (1 - 2z^{-1} + 4z^{-2}) \right] - 12 \left[\frac{1}{1 - z^{-1}} - (1 + z^{-1} + z^{-2}) \right] \end{aligned}$$

经整理得

$$X(z) = \frac{4z + 8}{(z - 1)(z + 1)(z - 2)^2}$$

做部分分式分解,并求 Z 反变换,注意到 $n \geq 3$,于是有

$$x(n) = \frac{4}{3}n \times 2^n - \frac{38}{9} \times 2^n + 6 + (-1)^n \frac{2}{9}$$

当 $n = M$ 时,有

$$M_R = \frac{4}{3}MN - \frac{38}{9}N + 6 + (-1)^M \frac{2}{9} \quad (4.5.7)$$

这正是分裂基算法在四类蝶形单元情况下所需的实数乘法次数的计算公式,同理可推出所需实数加法的递推公式为

$$A_R = \frac{8}{3}MN - \frac{16}{9}N + 2 - (-1)^M \frac{2}{9} \quad (4.5.8)$$

N \ 算法	基 2		基 4		基 2/4	
	M_R	A_R	M_R	A_R	M_R	A_R
2	0	4			0	4
4	0	16	0	16	0	16
8	4	52			4	52
16	28	148	24	144	24	144
64	332	964	264	920	248	912
256	2 316	5 380	1 800	5 080	1 656	5 008
1 024	13 324	27 652	10 248	25 944	9 336	25 488
4 096	69 644	135 172	53 256	126 926	48 248	123 792

各种算法所需计算量的比较

分裂基算法计算量推导

分裂基算法的特点

在序列长度满足 $N = 2^M$ 的各种算法中，分裂基算法所需乘法次数最少，接近理论最小值——实数乘法为 $M_R = 2N - M^2 - M - 2$;

有规则的结构、可以同址运算、便于IC设计;

分裂基算法较之于基2、基4算法，更加合理地安排了算法结构，最大程度地减小了无关紧要的旋转因子;

若考虑了所有无关紧要的旋转因子（包括 $\pm j$ ），那么三者的计算量其实都是一样的。

4.6 线性调频 Z 变换

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi nk}{N}} \quad k = 0, 1, \dots, N-1$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi nk}{N}} \quad n = 0, 1, \dots, N-1$$

DFT: 输入 N 点, 输出 N 点, 输入、输出点数相同。输出的 N 点均匀分布于单位圆上, **频域分辨率**为:

$$\frac{2\pi}{N} \Rightarrow \frac{f_s}{N}$$

在实际应用中：

1. 当输入点数极少时，若希望频率分点较多，则需要补零，结果是增加了计算量；
2. 对于窄带信号，我们只希望通带内分点密，带外可以较疏，或根本不用计算。

如何解决？

正弦
信号

解决方案

- 1. Pruning
- 2. CZT

线性调频 Z 变换：CZT

该名称来自于线性调频信号，即chirp信号

可以计算单位圆上某一段曲线上的z变换

计算DFT时，输入点数 N 和输出点数 M 可以不等，从而实现频域“细化”的目的。

Z变换: $X(z) = \sum_{n=0}^{\infty} x(n)z^{-n}$

其中 $z = e^{sT_s} = e^{(\sigma + j\Omega)T_s} = e^{\sigma T_s} e^{j\Omega T_s} = r e^{j\omega}$

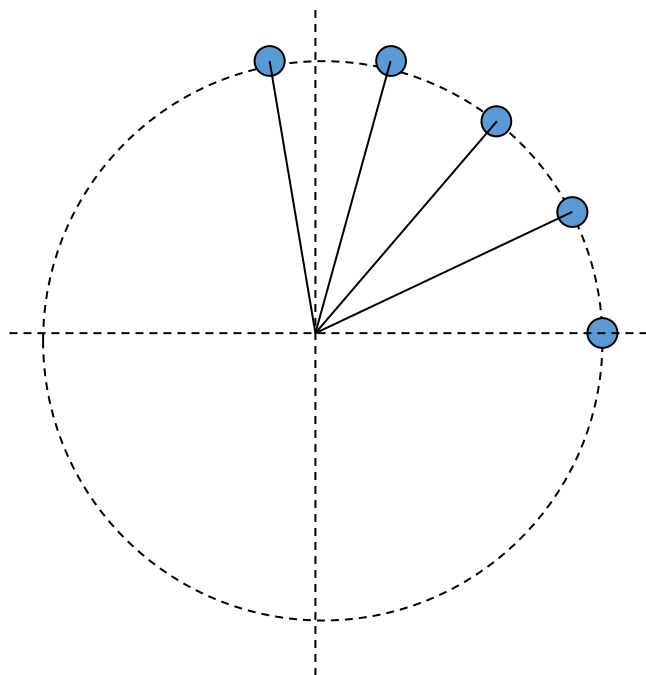
z 在其 ROC 内取值

现为Z指定一离散的路径:

$$z_r = AW^{-r}, \quad r = 0, 1, \dots, M-1$$
$$A = A_0 e^{j\theta_0}, \quad W = W_0 e^{-j\varphi_0}$$
$$z_r = A_0 e^{j\theta_0} W_0^{-r} e^{j\varphi_0 r}$$

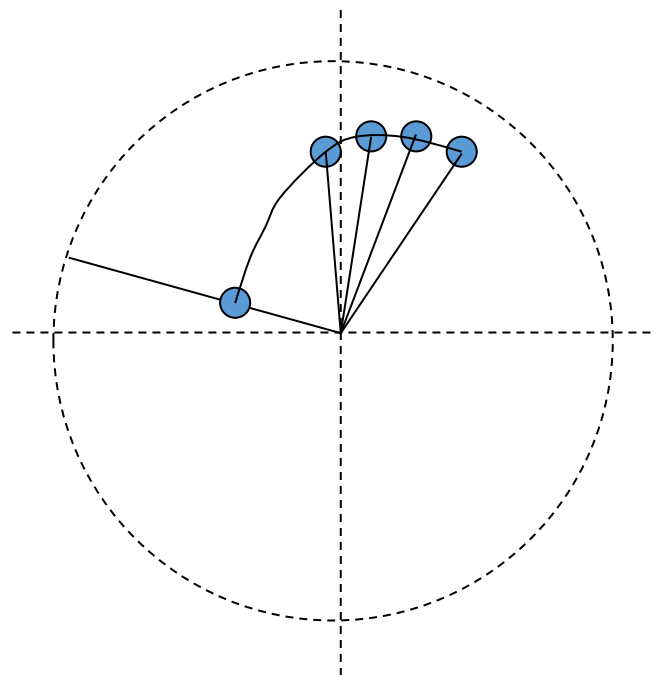
CZT: $x(n) \xrightarrow{\text{CZT}} X(z_r) = \text{CZT}[x(n)] = \sum_{n=0}^{\infty} x(n) z_r^{-n}$

$$= \sum_{n=0}^{\infty} x(n) A^{-n} W^{nr}$$



DFT变换路径

$$z_k = e^{j2\pi k/N}$$

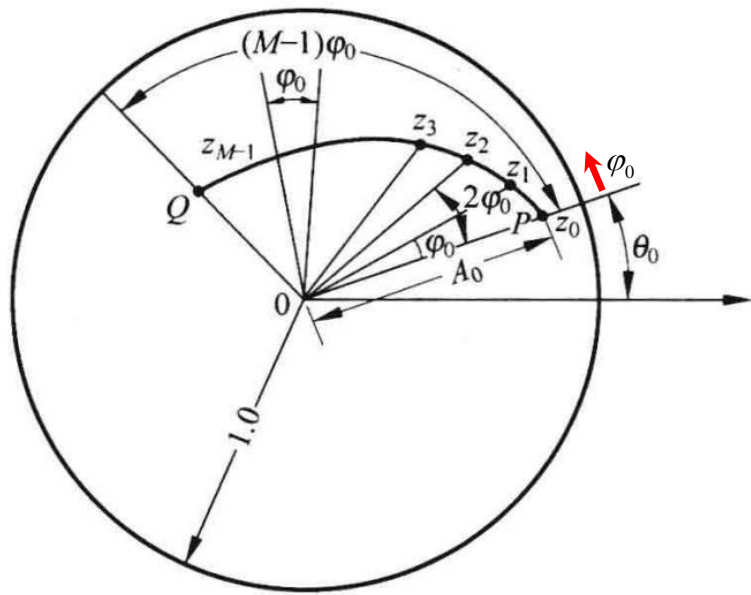


CZT变换路径

$$z_r = A_0 e^{j\theta_0} W_0^{-r} e^{j\varphi_0 r}$$

做 DFT 时，Z 变换在单位圆上的等分的 N 个点上取值

CZT 时，离散路径可在单位圆内、外，或圆上



CZT在Z平面上的变换，路径是一条螺旋线

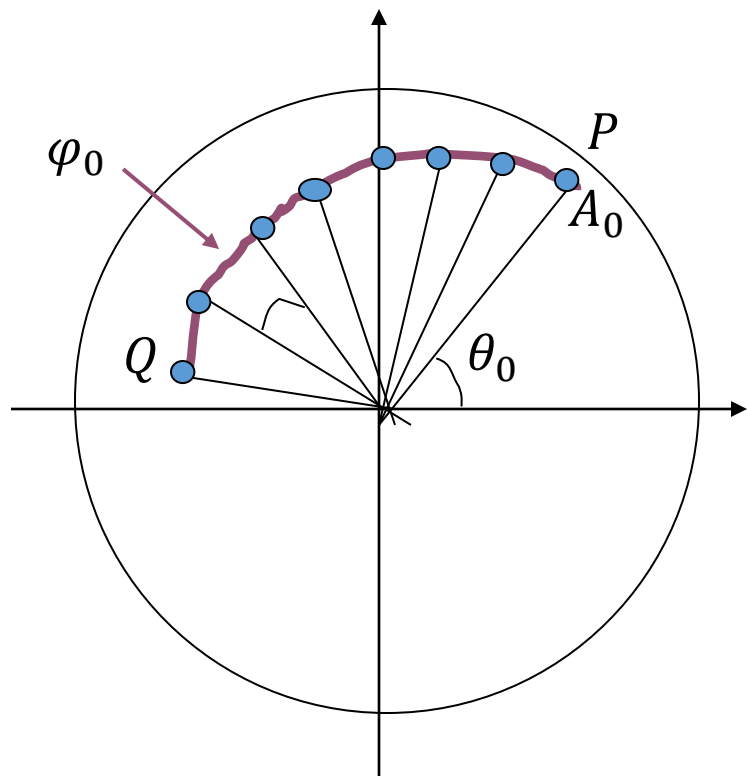
A_0, θ_0 决定CZT的起点 ($r = 0$)

W_0 决定变换路径如何倾斜，即影响到幅度的加权量 W_0^{-1}

φ_0 决定变换在角度上的增量

$$z_r = \underbrace{A_0 e^{j\theta_0}}_A \underbrace{W_0^{-r} e^{j\varphi_0 r}}_{W^{-r}}$$

信号的点数 N 和变换路径的点数 M 可以不相等



$A_0 > 1$ 时，起点在单位圆外，反之，在圆内；

$W_0 > 1$ 时，在 W_0^{-1} 的作用下，内旋，反之外旋；

$A_0 = W_0 = 1$ 时，CZT变换路径为单位圆上一段弧；

$$z_r = \underbrace{A_0 e^{j\theta_0}}_A \underbrace{W_0^{-r} e^{j\varphi_0 r}}_{W^{-r}}$$

$A_0 = W_0 = 1, \theta_0 = 0, M = N$ ，则CZT变成了DFT。

CZT的特点

- CZT可计算单位圆上任一段曲线上的 Z 变换, 可任意给定起止频率;
- 作变换时输入的点数 N 和输出点数 M 可以不相等;
- 可达到频域“细化”的目的。

用CZT做频谱分析：时间序列 N 点，频域序列 M 点

CZT的计算：
$$z_r = AW^{-r} = A_0 e^{j\theta_0} W_0^{-r} e^{j\varphi_0 r}$$

由定义：
$$X(z_r) = \sum_{n=0}^{N-1} x(n) z_r^{-n} = \sum_{n=0}^{N-1} x(n) A^{-n} W^{nr}$$

由于：
$$nr = \frac{1}{2} [r^2 + n^2 - (r-n)^2]$$

所以：
$$X(z_r) = \underbrace{W^{r^2/2}}_{\text{green}} \sum_{n=0}^{N-1} \underbrace{x(n) A^{-n} W^{n^2/2}}_{\text{blue}} \underbrace{W^{-(r-n)^2/2}}_{\text{dashed blue}}$$

对 n 和 r 取值范围的理解

令：
$$\begin{cases} g(n) = x(n) A^{-n} W^{n^2/2} \\ h(n) = W^{-n^2/2} \end{cases}$$

则：

$$X(z_r) = W^{r^2/2} \sum_{n=0}^{N-1} g(n)h(r-n)$$

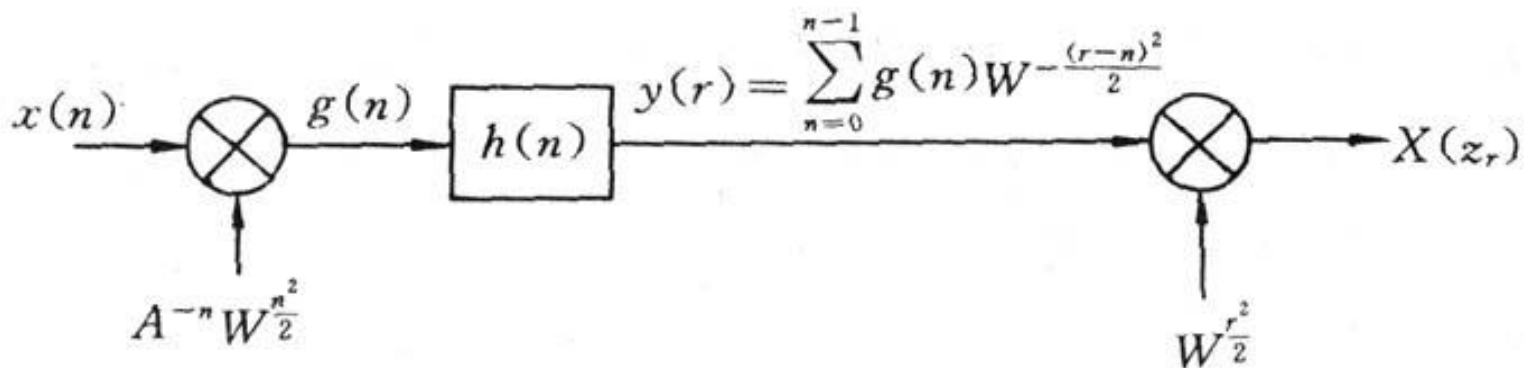
$$= W^{r^2/2} [g(r) * h(r)] = W^{r^2/2} y(r)$$

式中：

$$y(r) = g(r) * h(r) = \sum_{n=0}^{N-1} g(n)W^{-(r-n)^2/2},$$

$$r = 0, 1, \dots, M-1$$

CZT的线性滤波计算步骤？



CZT的实际计算方法：

$$y(r) = g(r) * h(r) = \sum_{n=0}^{N-1} g(n) W^{-(r-n)^2/2},$$
$$r = 0, 1, \dots, M-1$$

关键问题： 上式的快速计算？

基本特点：

1. $g(n)$ 是 N 点序列，由 $x(n)$ 所决定：

$$g(n) = x(n) A^{-n} W^{n^2/2}$$

2. $h(n)$ 是双边无穷长序列，由定义所决定：

$$h(n) = W^{-n^2/2}$$

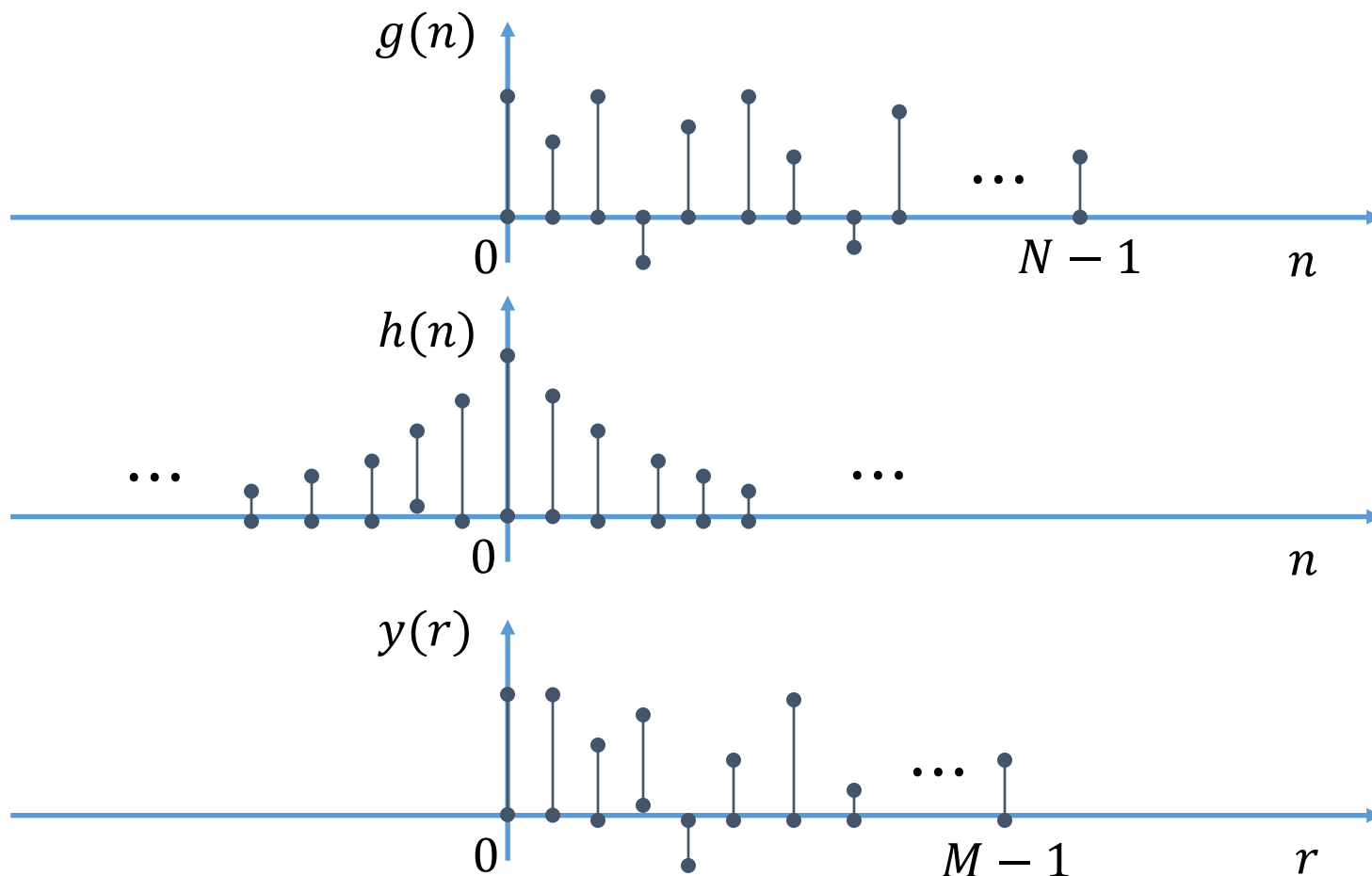
$g(n)$ 与 $h(n)$ 的卷积结果 $y(n)$ 为无穷序列： $-\infty < n < \infty$ ，算不完

3. $X(z_r)$ 是 M 点序列，由需要所决定。

——只需计算 M 点的卷积结果！

如何做？

不需要 $h(n)$ 的所有序列，可以对其截短。如何截短？

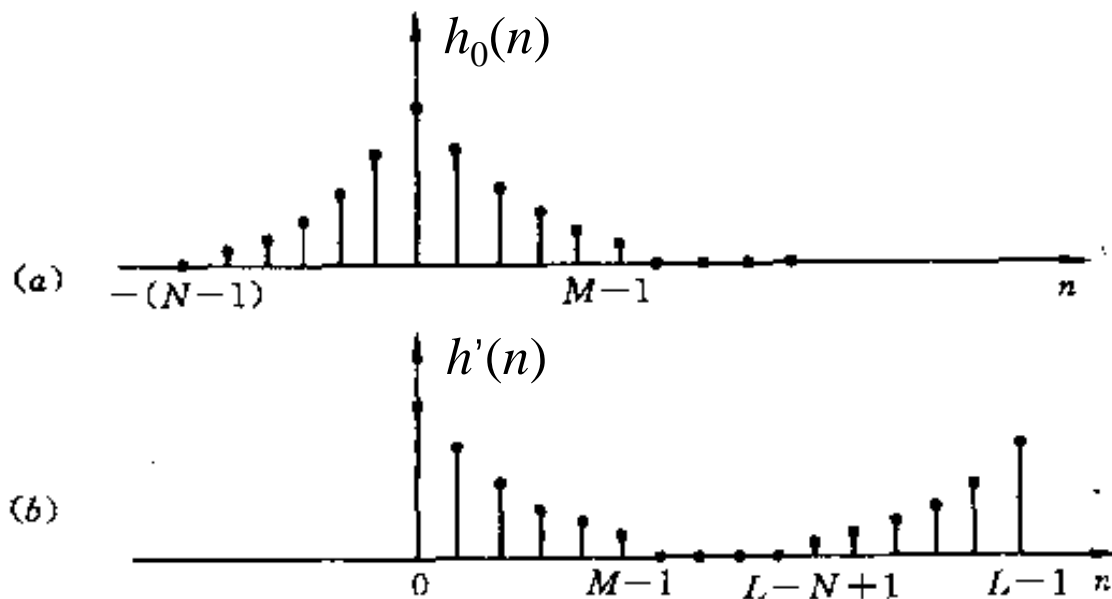


假定 $g(n)$ 不动，反转
平移 $h(n)$ ，根据计算
要求，可以确定出所
需要的 $h(n)$ 的左右边
界：

$$[-(N-1), M-1]$$

为方便，可记对应的
有限长序列为 $h_0(n)$

$$h(n) = W^{-n^2/2}$$



$h(n)$ 截短为 $h_0(n)$ ；补零且变换成能进行循环
卷积所需要的序列 $h'(n)$

用DFT计算上述线性卷积

圆周卷积计算线性卷积的补零分析

$g(r)$: $[0, N-1]$

$h_0(r)$: $[-(N-1), M-1]$

$y(r)$: $[-(N-1), N+M-2]$, 长度为 $2N+M-2$

按通常做法, 对 $g(n)$ 应补零个数为 $2N+M-2-N=N+M-2$, 对 $h_0(n)$ 应补零个数为?

对于 $y(r)$, 只需要 r 等于0到 $M-1$ 上的值。所以, 实际上对 $g(n)$ 只需补零个数为 $M-1$ 个, 可以少补零 $N-1$ 个。(?)

少补 $N-1$ 个零, 将在卷积结果的头部和尾部的 $N-1$ 个数据点上混叠的。而中间那些点上的值是不混叠的, 是想要的。

为了便于计算, 补零到长度为 L , 且 L 满足FFT计算的需要

具体计算步骤

Step 1.

$$h'(n) = \begin{cases} h(n) & 0 \leq n \leq M-1 \\ 0 & M \leq n \leq L-N \\ h(L-n) & L-N+1 \leq n \leq L-1 \end{cases}$$

如果 $g(n)$ 只补零 $M-1$ 个，则 $g(n)$ 的长度为 $N+M-1$ ，而 $h_0(n)$ 的长度正好是 $N+M-1$ ，故它无需补零。

也可以对 $h_0(n)$ 补零，则出现上式中的0值区间！当然对应地要多给 $g(n)$ 再补相应个数的零。

Step 2. 求出 $g'(n)$, $n = 0, 1, \dots, N - 1$

$$g'(n) = \begin{cases} g(n) & n = 0, 1, \dots, N - 1 \\ 0 & N \leq n \leq L - 1 \end{cases}$$

$g'(n), h'(n)$: L 点序列

Step 3. $g'(n), h'(n) \Rightarrow H'(k), G'(k)$

Step 4. $Y'(k) = H'(k)G'(k), \Rightarrow y'(r)$

Step 5. $X(z_r) = y(r)W^{r^2/2}$

与本章内容有关的MATLAB文件

与本章内容有关的MATLAB文件主要是fft, ifft和czt.m。

fft实现快速傅立叶变换，ifft实现快速傅立叶反变换，czt.m 用来实现线性调频Z变换。

1. fft的调用格式是： $X=\text{fft}(x)$, 或 $X=\text{fft}(x, N)$ 。
2. czt.m 调用格式是： $X=\text{czt}(x, M, W, A)$ 。 x 是待变换的时域信号，其长度设为 N ， M 是变换的长度， W 确定变换的步长， A 确定变换的起点。若 $M=N, A=1$, 则CZT变成DFT。