100052205

数字信号处理

Digital Signal Processing

李慧琦 教授

信息与电子学院

北京理工大学

Tel: +86 (10) 68918239

Email: huiqili@bit.edu.cn

第四章 快速傅里叶变换（FFT）

本章主要内容

- 快速计算DFT的基本思路
- 基2按时间抽取FFT算法
- 基2按频率抽取FFT算法
- N为复合数的FFT方法
- 分裂基FFT算法
- Chirp-Z 变换
- FFT的应用：实序列FFT算法、卷积、相关计算

§ 4-5 N为复合数的FFT算法 —统一的FFT算法

$N = 2^{\nu} \rightarrow$ 基$-2\ \ FFT$

$N \neq 2^{\nu}$，如何快速计算 DFT？

处理方法：

(1)通过补零，使序列长度=2^{ν} → 基-2 FFT

(2)N=PQ（复合数） → 统一的FFT算法（基**2**算法的扩展）

(3)N≠PQ（素数） → Chirp-Z 变换(CZT)

一、算法原理

$$\forall x(n), \quad 0 \le n \le N-1, \quad N = PQ$$（复合数）

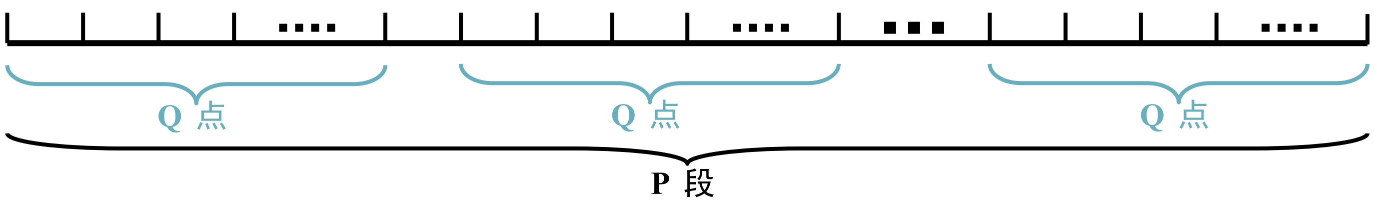
$\because$ *N-DFT~N²*

$$\therefore \text{如果}N\text{-}DFT \begin{cases} Q\text{个}P\text{-}DFT \sim Q \times P^2 \\ P\text{个}Q\text{-}DFT \sim P \times Q^2 \end{cases} \longrightarrow \text{减少了运算}$$

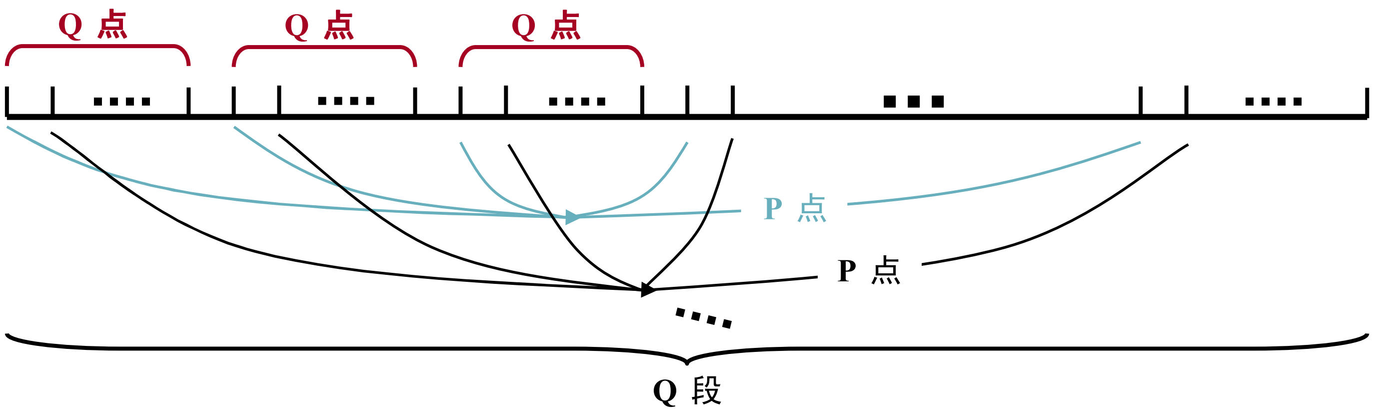
Input:

$N = P \times Q$

- $n = Qn_1 + n_0,\ n_1 = 0 \sim P-1;\ n_0 = 0 \sim Q-1$
- n_1固定, n_0变化（分段）：计分成 P 段，每段由毗邻的 Q 点组成
- n_0固定, n_1变化（抽取）：计分成 Q 段，每段由距离 Q 点的 P 点组成



分段：
可得 P 个 Q 点序列

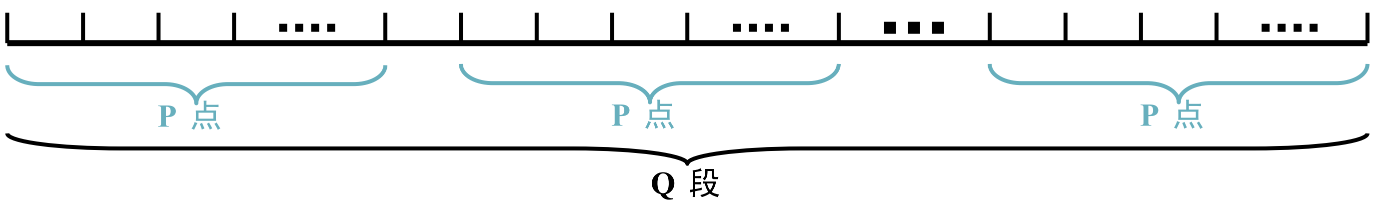


抽取：
可得 Q 个 P 点序列

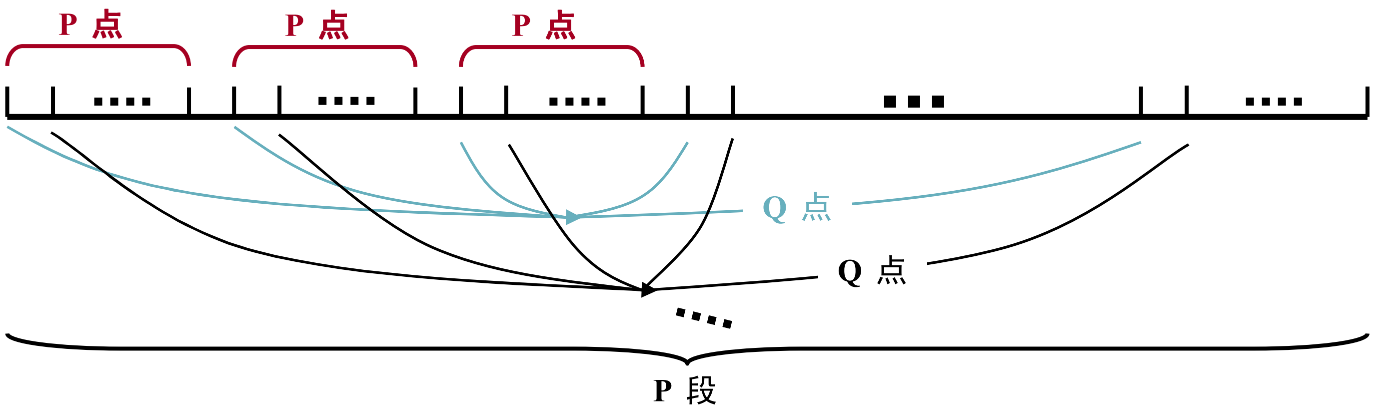
Output:

$N = P\times Q$

- $k = Pk_1 + k_0$, $k_1 = 0 \sim Q-1$; $k_0 = 0 \sim P-1$
- k_1固定, k_0变化（分段）：计分成 Q 段，每段由毗邻的 P 点组成
- k_0固定, k_1变化（抽取）：计分成 P 段，每段由距离 P 点的 Q 点组成



分段：
可得 Q 个 P 点序列

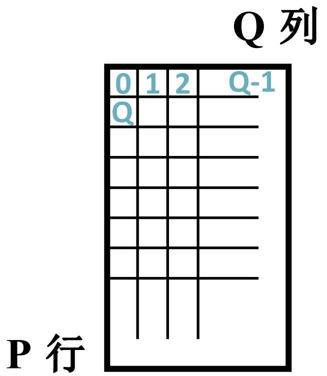


抽取：
可得 P 个 Q 点序列

以时域抽取，频域分段为例：

- $n = Qn_1 + n_0 \,,\; n_1 = 0 \sim P - 1 \;;\; n_0 = 0 \sim Q - 1$

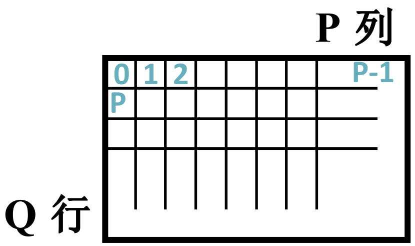
抽取：
可得 Q 个 P 点序列



- 矩阵各列正好是抽取得到的 Q 个 P 点序列
- $x(n) = x(n_1, n_0)$

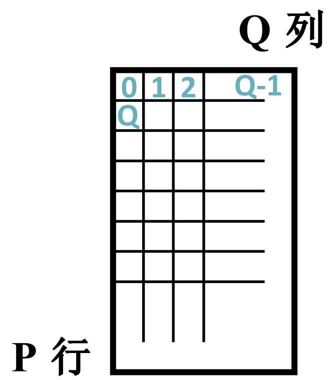
- $k = Pk_1 + k_0 \,,\; k_1 = 0 \sim Q - 1 \;;\; k_0 = 0 \sim P - 1$

分段：
可得 Q 个 P 点序列



- 矩阵各行正好是分段得到的 Q 个 P 点序列
- $X(k) = X(k_1, k_0)$

算法推导:



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

$$=\sum_{n_0=0}^{Q-1}\sum_{n_1=0}^{P-1}x(Qn_1+n_0)e^{-j\frac{2\pi}{N}k(Qn_1+n_0)}\ ,\ k=0\sim N-1$$

$$=\sum_{n_0=0}^{Q-1}\sum_{n_1=0}^{P-1}x(Qn_1+n_0)e^{-j\frac{2\pi}{N}kn_0}e^{-j\frac{2\pi}{N}kQn_1}\ ,\ k=0\sim N-1$$

$$=\sum_{n_0=0}^{Q-1}\underbrace{\left\{\sum_{n_1=0}^{P-1}\left[x(Qn_1+n_0)e^{-j\frac{2\pi}{N}kn_0}\right]e^{-j\frac{2\pi}{P}kn_1}\right\}}_{DFT\left\{x(Qn_1+n_0)e^{-j\frac{2\pi}{N}kn_0}\right\}}\ ,\ k=0\sim N-1$$

时域:对于某固定n_0, $n_1=0\sim P-1$
频域:$k=0\sim P-1$

频域需分段处理:$k=Pk_1+k_0$
$k_1=0\sim Q-1\ ;\ k_0=0\sim P-1$

$$X(Pk_1+k_0)=\sum_{n_0=0}^{Q-1}\left\{\sum_{n_1=0}^{P-1}\left[x(Qn_1+n_0)e^{-j\frac{2\pi}{N}(Pk_1+k_0)n_0}\right]e^{-j\frac{2\pi}{P}(Pk_1+k_0)n_1}\right\}$$

$$=\sum_{n_0=0}^{Q-1}\left\{\sum_{n_1=0}^{P-1}\left[x(Qn_1+n_0)e^{-j\frac{2\pi}{N}Pk_1n_0}e^{-j\frac{2\pi}{N}k_0n_0}\right]e^{-j\frac{2\pi}{P}Pk_1n_1}e^{-j\frac{2\pi}{P}k_0n_1}\right\}$$

$$=\sum_{n_0=0}^{Q-1}\left\{\underbrace{\left[\sum_{n_1=0}^{P-1}x(Qn_1+n_0)e^{-j\frac{2\pi}{P}k_0n_1}\right]}_{}e^{-j\frac{2\pi}{N}k_0n_0}\right\}e^{-j\frac{2\pi}{Q}k_1n_0}$$

$$X_1(k_0,n_0)=DFT\{x(Qn_1+n_0)\},n_0=0\sim Q-1;k_0=0\sim P-1$$

Q 个 P 点 DFT 输出

$$X_1'(k_0,n_0)=X_1(k_0,n_0)W_N^{k_0n_0},n_0=0\sim Q-1;k_0=0\sim P-1$$

QP 个乘法

$$X_2(k_0,k_1)=DFT\{X_1'(k_0,n_0)\},k_1=0\sim Q-1;k_0=0\sim P-1$$

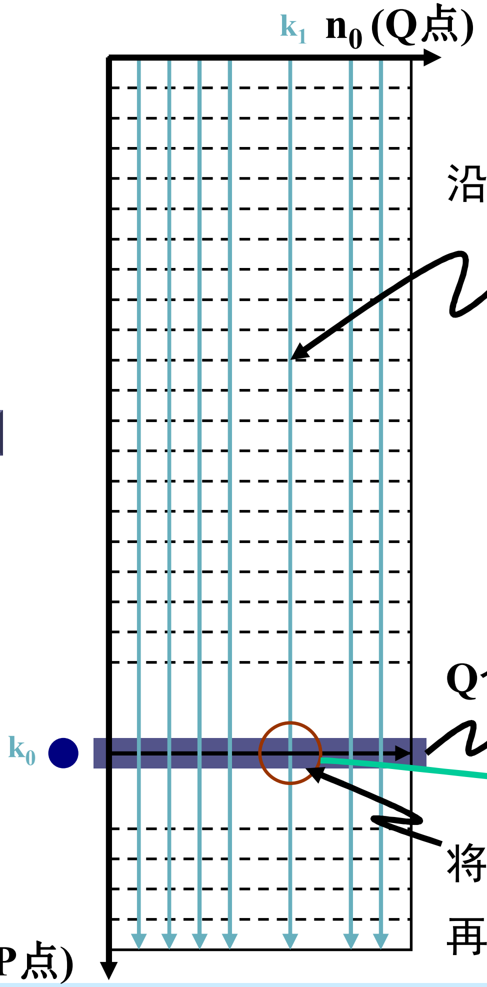
P 个 Q 点 DFT 输出

MR-FFT图解

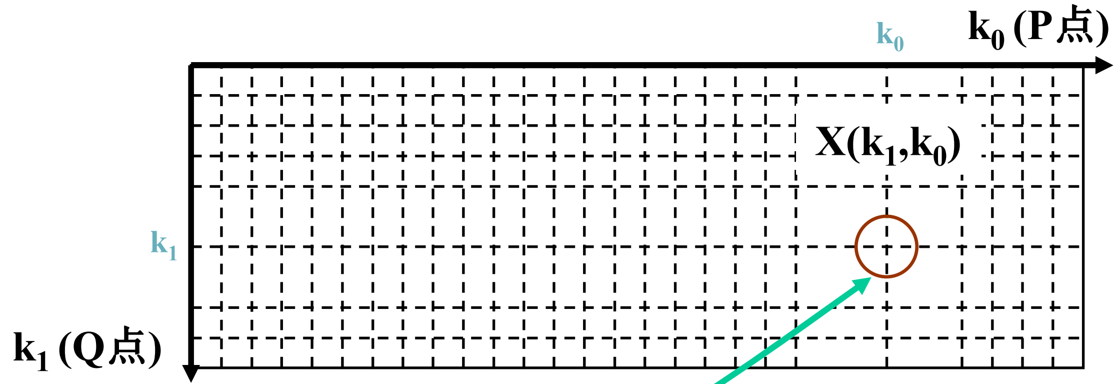
乘法次数： $QP^2 + QP + PQ^2 = (P+Q+1)N$

实现步骤：Q个P点DFT->序列加权->P个Q点DFT

P Q Q P



沿着n_0轴每一列都作P点DFT

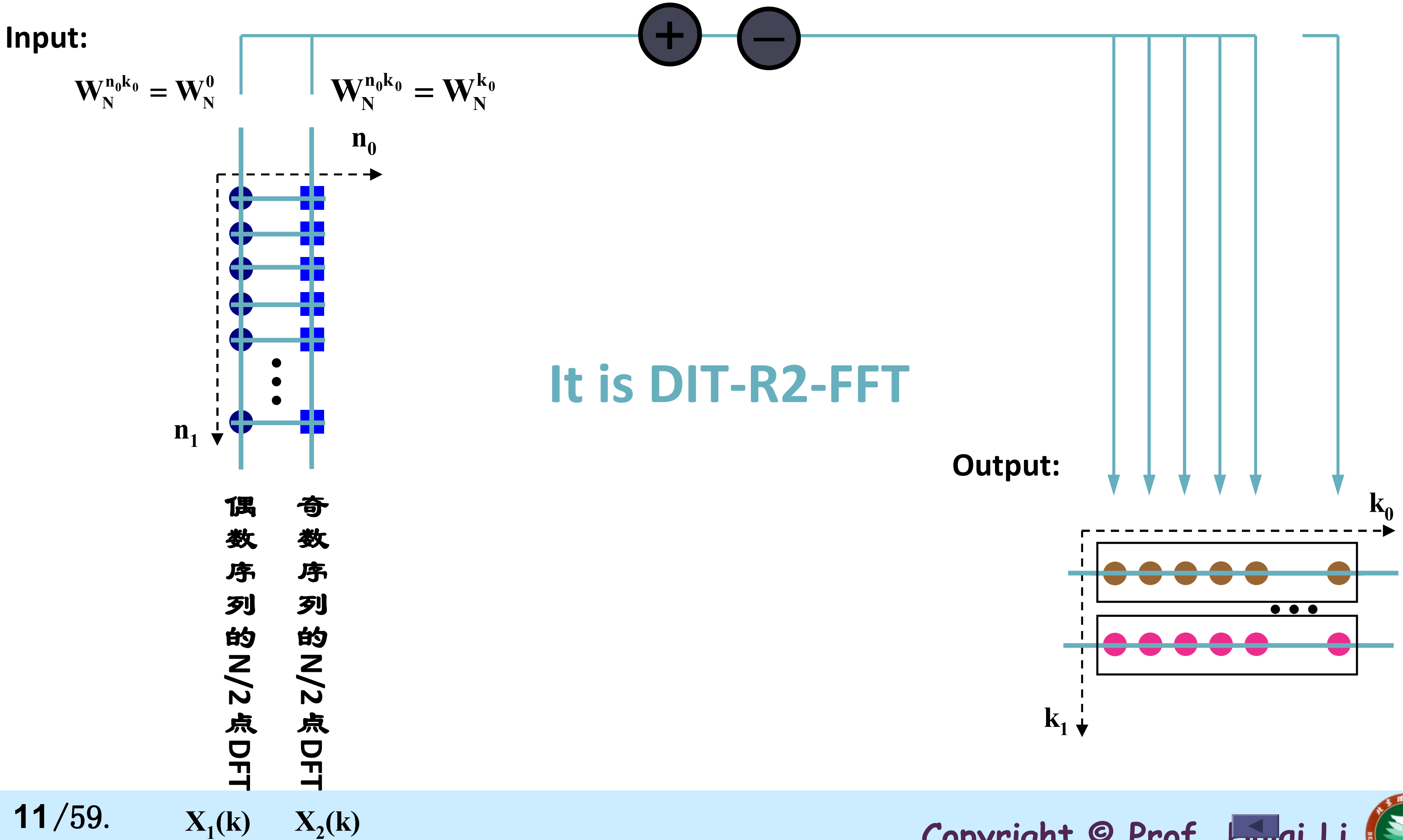


Q个P点DFT的第k_0点组成新的Q点序列 $X_1(k_0,n_0)=y_{k_0}(n_0)$

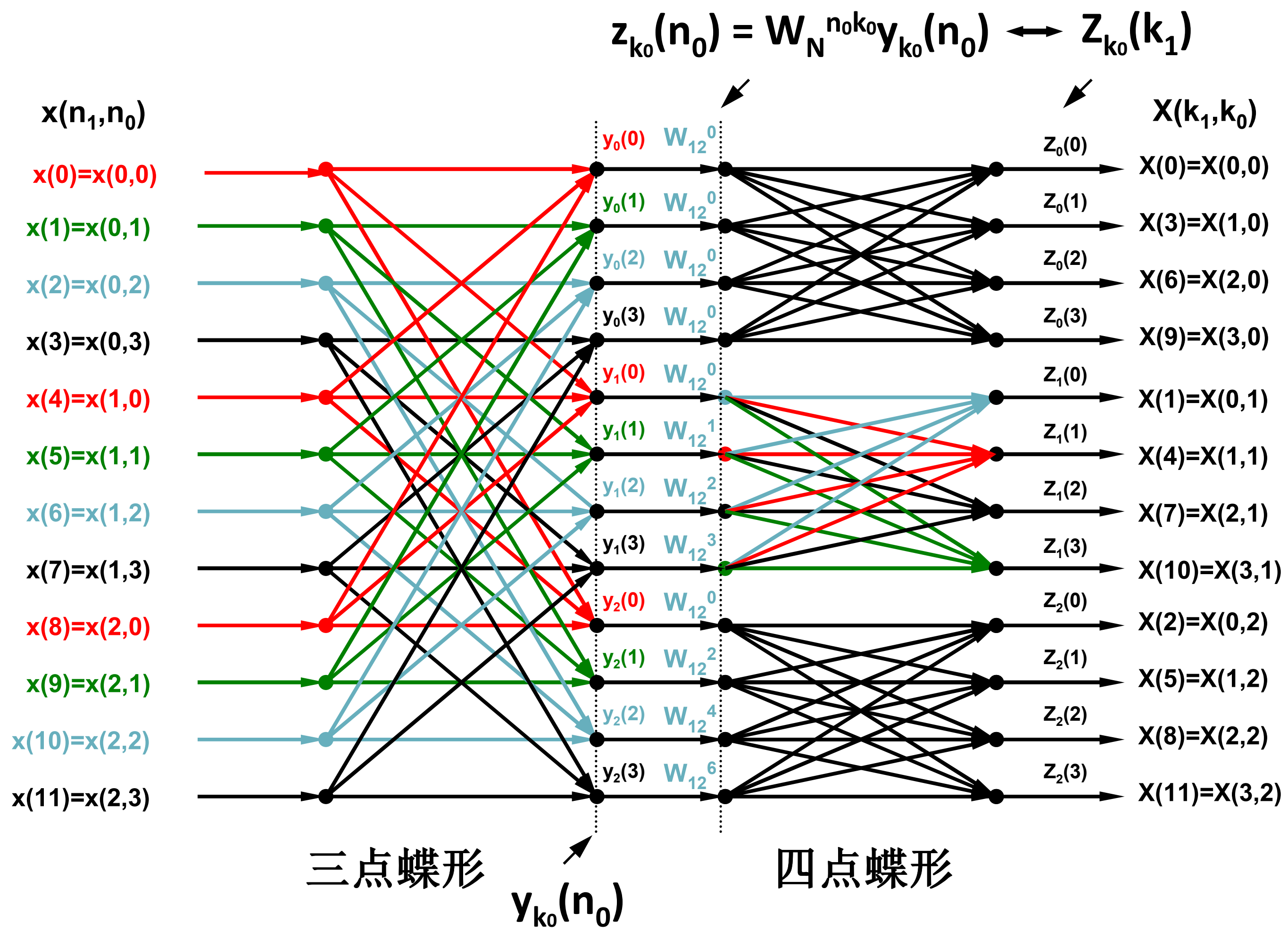
将 $y_{k_0}(n_0)$ 乘以 $W_N^{k_0n_0}$，得到 $z_{k_0}(n_0)=W_N^{k_0n_0}y_{k_0}(n_0)$

再作Q点DFT，第k_1点输出即是 $Z_{k0}(k_1)=X(k_1,k_0)$

what happens when P = N/2, Q = 2 ？



It is DIT-R2-FFT



乘法次数：96 < 144

**12点复合基FFT算法第一次分解示意图

二、运算步骤

(1) $\boldsymbol{x(n) \rightarrow x(n_1, n_0)}$

$\uparrow$

$\boldsymbol{n = Q n_1 + n_0}$

$n_1 = 0,1,\ldots,P-1$ 行号

$n_0 = 0,1,\ldots,Q-1$ 列号

(2) $\forall n_0, \quad 0 \le n_0 \le Q-1$ （针对每一列）

$$X_1(k_0, n_0) = DFT_{n_1}[x(n_1, n_0)] = \sum_{n_1=0}^{P-1} x(n_1, n_0) W_L^{k_0 n_1}, \qquad k_0 = 0,1,\ldots,L-1$$

(3) $\boldsymbol{X_1'(k_0, n_0) = X_1(k_0, n_0) W_N^{k_0 n_0} \quad 0 \le k_0 \le P-1, \quad 0 \le n_0 \le Q-1}$

(4) $\forall k_0, \quad 0 \le k_0 \le P-1$ （针对每一行）

$$X_2(k_0, k_1) = DFT_{n_0}[X_1'(k_0, n_0)] = \sum_{n_0=0}^{Q-1} X_1'(k_0, n_0) W_P^{k_1 n_0}, \qquad k_0 = 0,1,\ldots,Q-1$$

(5) 译序

$$X_2(k_0, k_1) \rightarrow X(k_1, k_0) \rightarrow X(k) \qquad 0 \le k \le N-1$$

$\uparrow$ $\qquad 0 \le k_0 \le P-1$

$k = P k_1 + k_0, \qquad 0 \le k_1 \le Q-1$

先行后列， 与书中顺序相反

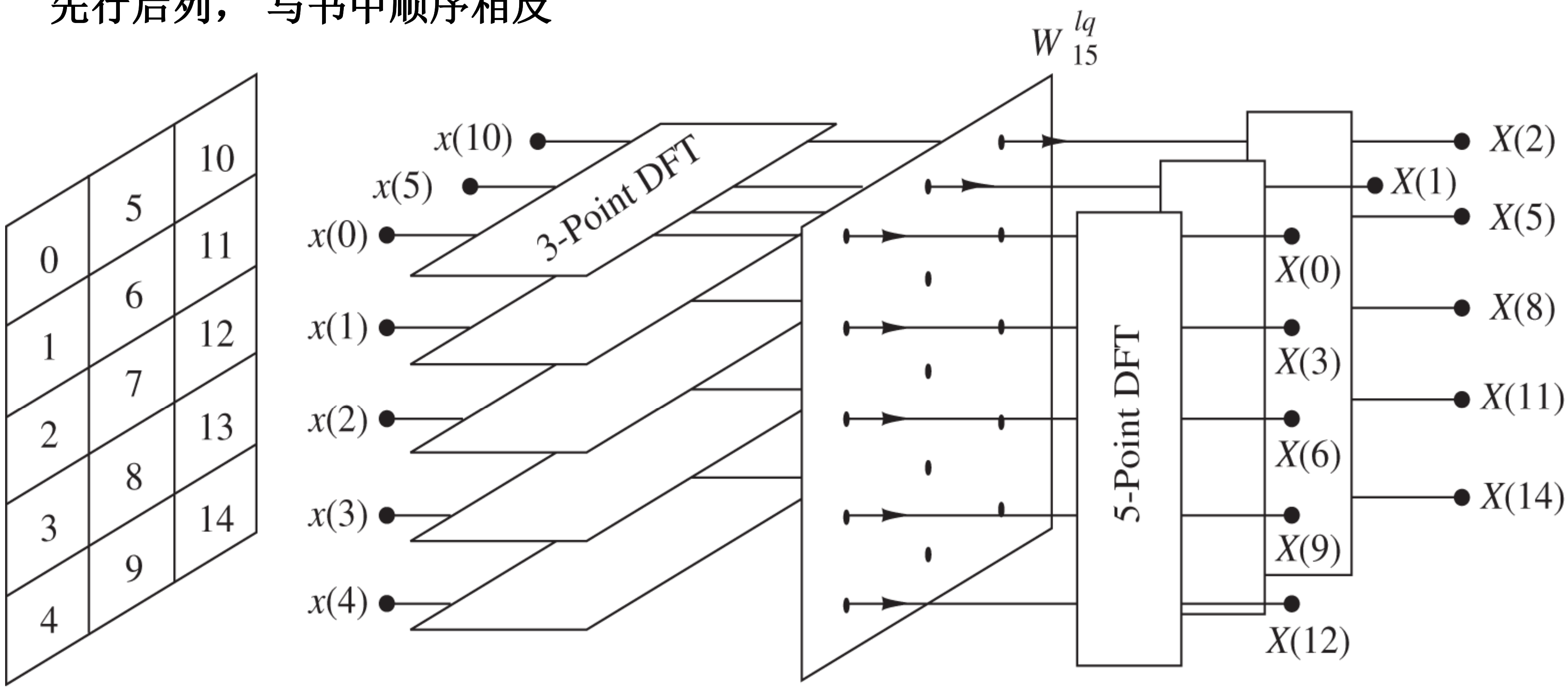


Figure 8.1.3 Computation of $N = 15$-point DFT by means of 3-point and 5-point DFTs.

三、基数（指特定的分解）

1. $N=2^{\nu}$ →基2 FFT算法

2. $N\neq 2^{\nu}$

(1) $N=r_1,r_2,\ldots,r_M$

M级$r_1,r_2,\ldots, r_M$点DFT →混合基算法

(2) $r_1=r_2=\ldots=r_M$ →$N= r^M$

M级r-DFT → 基-r FFT算法

比如：a) $N=2^M$ →基-2 FFT

b) $N=4^M$ →基-4 FFT

四、运算量估算

$N=PQ$

(1) Q个**P**-DFT：$\times$— $Q\times P^2=N\times P$

$+$— $Q\times P(P-1)=N(P-1)$

(2) 乘N个$W_N^{k_0 n_0}$因子：$\times$— N

(3) P个Q-DFT：$\times$— $P\times Q^2=N\times Q$

$+$— $P\times Q(Q-1)=N(Q-1)$

总运算量：$\times$— $NP+N+NQ=N(P+Q+1) < N^2$

$+$— $N(P-1)+N(Q-1)=N(P+Q-2) < N(N-1)$

§4-6 分裂基FFT算法(Split-Radix FFT)

一、背景

对更快速算法的需求

$$基-2\,FFT \sim \frac{N}{2}\log_2^N$$

$$\downarrow$$

1984年，杜梅尔（P.Douhamel）

霍尔曼（H.Hollman）

基-2 FFT

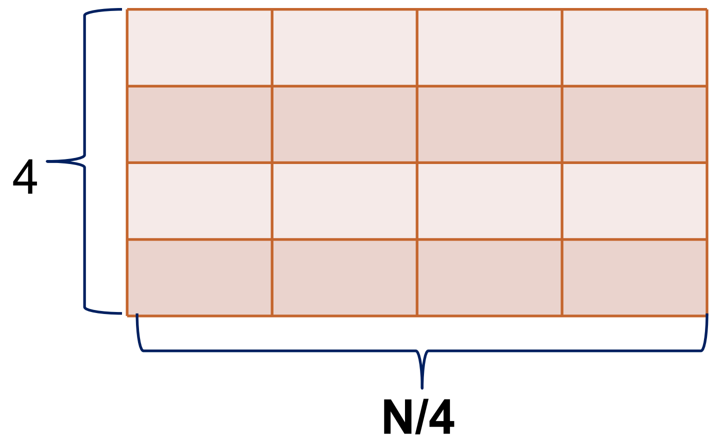
基-4 FFT

$$\longrightarrow 分裂基\,FFT \sim \frac{N}{3}\log_2^N$$

1. 最少乘法

2. 原位运算

基4- FFT算法 （DIF）



$\forall x(n), \quad 0 \le n \le N-1, \quad N = 4^{\nu}$

设 $N=Pq, \quad P=N/4, \quad q=4$

令 $n = Pn_1 + n_0 = N/4\, n_1 + n_0$

$0 \leqslant n_1 \leqslant 3, \qquad 0 \leqslant n_0 \leqslant (N/4)-1$

$k = 4k_1 + k_0$

$0 \leqslant k_1 \leqslant (N/4)-1, \quad 0 \leqslant k_0 \leqslant 3$

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

$$= \sum_{n_0=0}^{N/4-1} \sum_{n_1=0}^{3} x(\frac{N}{4} n_1 + n_0) W_N^{k(\frac{N}{4} n_1 + n_0)}$$

$$= \sum_{n_0=0}^{N/4-1} \left[x(n_0) W_4^0 + x(n_0 + \frac{N}{4}) W_4^k + x(n_0 + \frac{N}{2}) W_4^{2k} + x(n_0 + \frac{3N}{4}) W_4^{3k} \right] W_N^{kn_0}$$

$$X(k) = X(4k_1 + k_0)$$

$$= \sum_{n_0=0}^{N/4-1} \left[x(n_0) + x(n_0 + \frac{N}{4}) W_4^{(4k_1+k_0)} + x(n_0 + \frac{N}{2}) W_4^{2(4k_1+k_0)} + x(n_0 + \frac{3N}{4}) W_4^{3(4k_1+k_0)} \right] W_N^{4(4k_1+k_0)n_0}$$

$$= \sum_{n_0=0}^{N/4-1} \left[x(n_0) + x(n_0 + \frac{N}{4}) W_4^{k_0} + x(n_0 + \frac{N}{2}) W_4^{2k_0} + x(n_0 + \frac{3N}{4}) W_4^{3k_0} \right] W_N^{4(4k_1+k_0)n_0}$$

在$k_0=0,1,2,3$时，用k表示k_1，n表示n_0

$$
\begin{cases}
X(4k)=\sum_{n=0}^{N/4-1}\left[x(n)+x(n+\frac{N}{4})+x(n+\frac{N}{2})+x(n+\frac{3N}{4})\right]W_N^{4kn} \\
X(4k+1)=\sum_{n=0}^{N/4-1}\left[x(n)-jx(n+\frac{N}{4})-x(n+\frac{N}{2})+jx(n+\frac{3N}{4})\right]W_N^{4kn+n} \\
X(4k+2)=\sum_{n=0}^{N/4-1}\left[x(n)-x(n+\frac{N}{4})+x(n+\frac{N}{2})-x(n+\frac{3N}{4})\right]W_N^{4kn+2n} \\
X(4k+3)=\sum_{n=0}^{N/4-1}\left[x(n)+jx(n+\frac{N}{4})-x(n+\frac{N}{2})-jx(n+\frac{3N}{4})\right]W_N^{4kn+3n}
\end{cases}
$$

$\downarrow$

基 4-DIF

$$0\le k\le\frac{N}{4}-1$$

➢ 基4 DIF另一种推导方法

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

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

$$=\sum_{n=0}^{N/4-1}x(n)W_N^{kn}+W_N^{Nk/4}\sum_{n=0}^{N/4-1}x(n+\frac{N}{4})W_N^{kn}$$

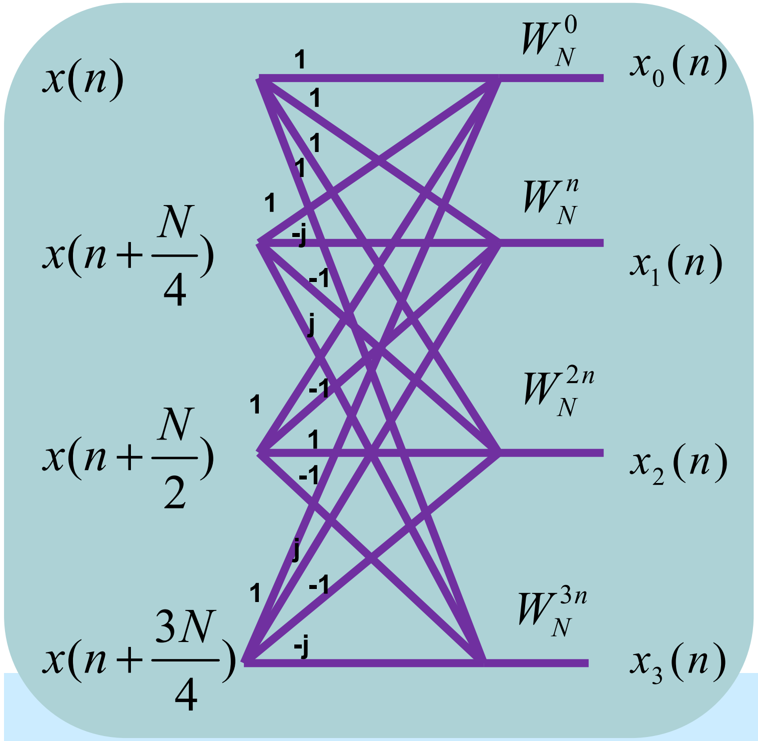
$$+W_N^{Nk/2}\sum_{n=0}^{N/4-1}x(n+\frac{N}{2})W_N^{kn}+W_N^{3Nk/4}\sum_{n=0}^{N/4-1}x(n+\frac{3N}{4})W_N^{kn}$$

$$W_N^{kN/4}=(-j)^k,\qquad W_N^{kN/2}=(-1)^k,\qquad W_N^{3kN/4}=(j)^k$$

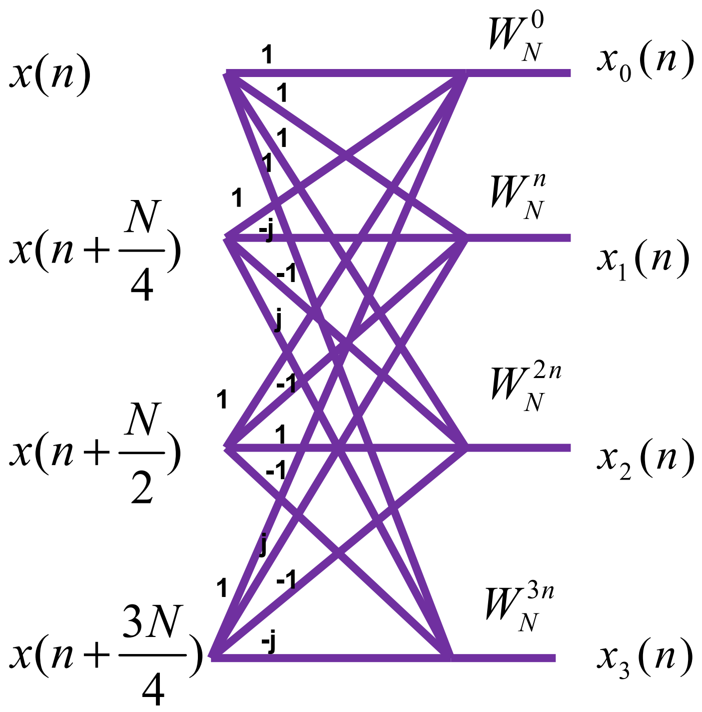
$$X(k)=\sum_{n=0}^{N/4-1}\left[x(n)+(-j)^k x(n+\frac{N}{4})+(-1)^k x(n+\frac{N}{4})+(j)^k x(n+\frac{3N}{4})\right]W_N^{nk}$$

$$\left\{\begin{aligned}
X(4k) &= \sum_{n=0}^{N/4-1}\left[x(n)+x(n+\frac{N}{4})+x(n+\frac{N}{2})+x(n+\frac{3N}{4})\right]W_N^0 W_{N/4}^{kn}\\
X(4k+1) &= \sum_{n=0}^{N/4-1}\left[x(n)-jx(n+\frac{N}{4})-x(n+\frac{N}{2})+jx(n+\frac{3N}{4})\right]W_N^n W_{N/4}^{kn}\\
X(4k+2) &= \sum_{n=0}^{N/4-1}\left[x(n)-x(n+\frac{N}{4})+x(n+\frac{N}{2})-x(n+\frac{3N}{4})\right]W_N^{2n} W_{N/4}^{kn}\\
X(4k+3) &= \sum_{n=0}^{N/4-1}\left[x(n)+jx(n+\frac{N}{4})-x(n+\frac{N}{2})-jx(n+\frac{3N}{4})\right]W_N^{3n} W_{N/4}^{kn}
\end{aligned}\right.$$

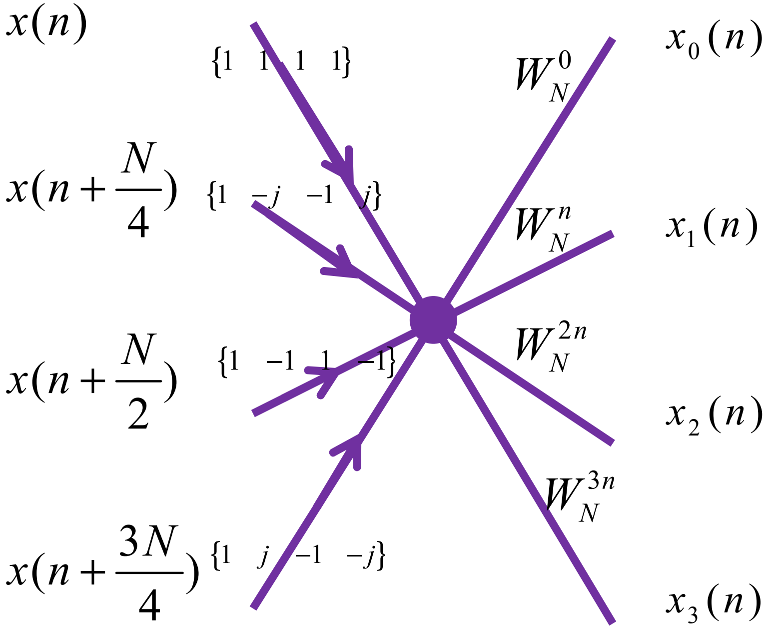
$$0 \le k \le \frac{N}{4}-1$$



$$\left\{\begin{aligned}
x_0(n) &= \left[x(n)+x(n+\frac{N}{4})+x(n+\frac{N}{2})+x(n+\frac{3N}{4})\right]W_N^0\\
x_1(n) &= \left[x(n)-jx(n+\frac{N}{4})-x(n+\frac{N}{2})+jx(n+\frac{3N}{4})\right]W_N^n\\
x_2(n) &= \left[x(n)-x(n+\frac{N}{4})+x(n+\frac{N}{2})-x(n+\frac{3N}{4})\right]W_N^{2n}\\
x_3(n) &= \left[x(n)+jx(n+\frac{N}{4})-x(n+\frac{N}{2})-jx(n+\frac{3N}{4})\right]W_N^{3n}
\end{aligned}\right.$$



$x(n)$
$x(n+\frac{N}{4})$
$x(n+\frac{N}{2})$
$x(n+\frac{3N}{4})$
W_N^0
W_N^n
W_N^{2n}
W_N^{3n}
$x_0(n)$
$x_1(n)$
$x_2(n)$
$x_3(n)$



$x(n)$
$\{1\ \ 1\ \ 1\ \ 1\}$
$x(n+\frac{N}{4})$
$\{1\ \ -j\ \ -1\ \ j\}$
$x(n+\frac{N}{2})$
$\{1\ \ -1\ \ 1\ \ -1\}$
$x(n+\frac{3N}{4})$
$\{1\ \ j\ \ -1\ \ -j\}$
W_N^0
W_N^n
W_N^{2n}
W_N^{3n}
$x_0(n)$
$x_1(n)$
$x_2(n)$
$x_3(n)$

4-DFT：

$$X(k)=\sum_{n=0}^{3}x(n)W_4^{kn}$$

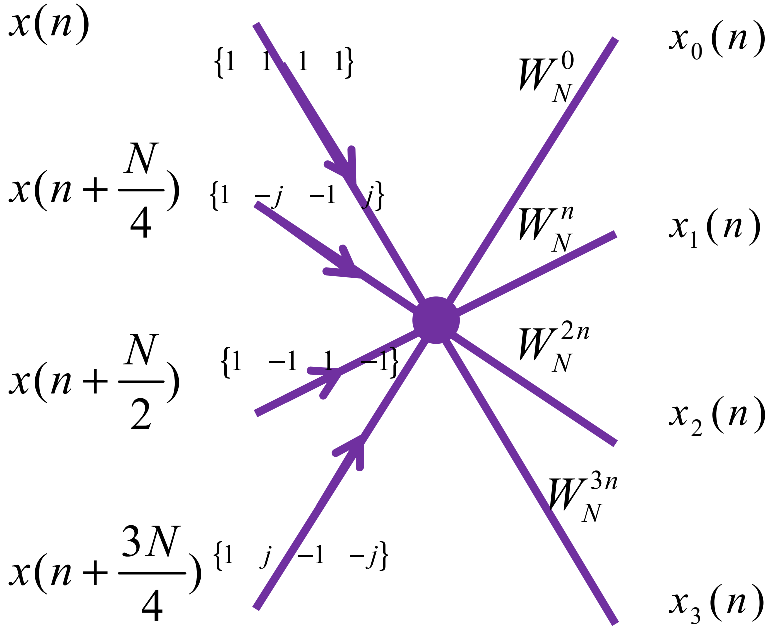
$$=x(0)W_4^0+x(1)W_4^k+x(2)W_4^{2k}+x(3)W_4^{3k}\qquad k=0,1,2,3$$

$$X(0)=x(0)+x(1)+x(2)+x(3)$$

$$X(1)=x(0)-jx(1)-x(2)+jx(3)$$

$$X(2)=x(0)-x(1)+x(2)-x(3)$$

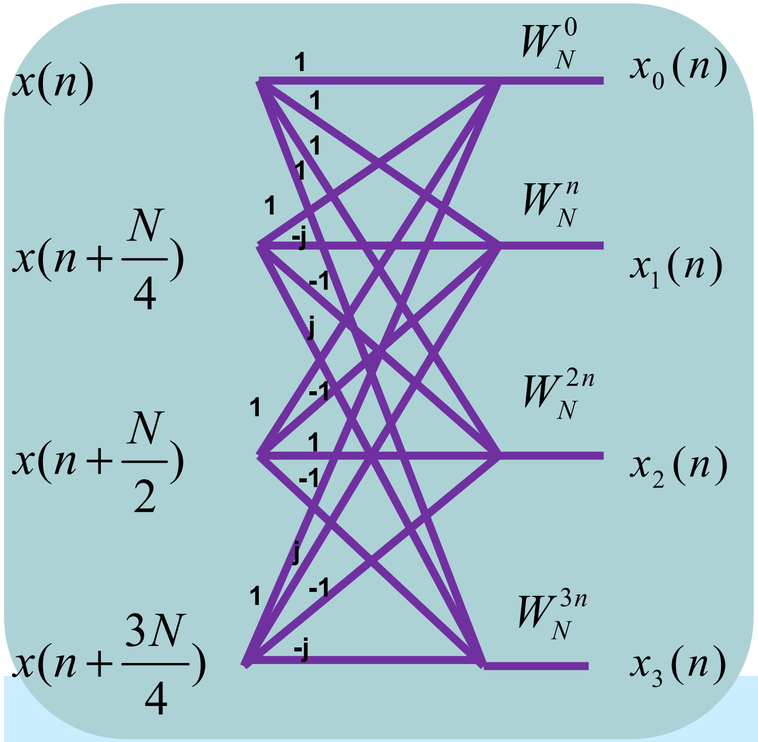
$$X(3)=x(0)+jx(1)-x(2)-jx(3)$$

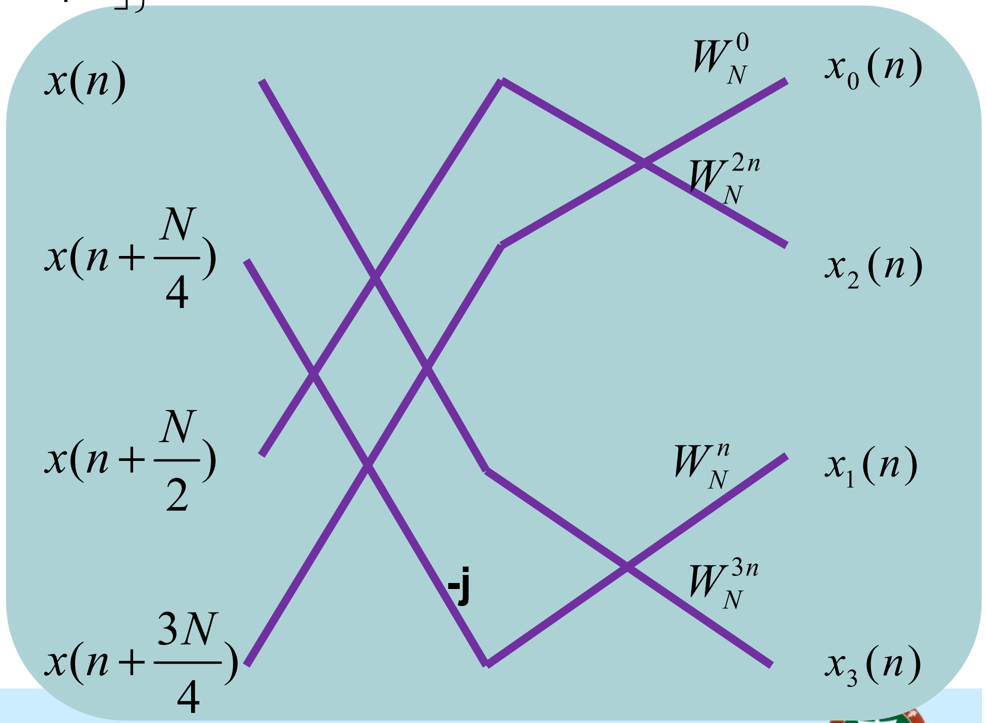


$$
\begin{cases}
x_0(n) = \left\{\left[x(n) + x(n+\frac{N}{2})\right] + \left[x(n+\frac{N}{4}) + x(n+\frac{3N}{4})\right]\right\} W_N^0 \\
x_1(n) = \left\{\left[x(n) - x(n+\frac{N}{2})\right] - j\left[x(n+\frac{N}{4}) - x(n+\frac{3N}{4})\right]\right\} W_N^n \\
x_2(n) = \left\{\left[x(n) + x(n+\frac{N}{2})\right] - \left[x(n+\frac{N}{4}) + x(n+\frac{3N}{4})\right]\right\} W_N^{2n} \\
x_3(n) = \left\{\left[x(n) - x(n+\frac{N}{2})\right] + j\left[x(n+\frac{N}{4}) - x(n+\frac{3N}{4})\right]\right\} W_N^{3n}
\end{cases}
$$

蝶形运算：×— 3次
+— 8次

4点DFT基2算法





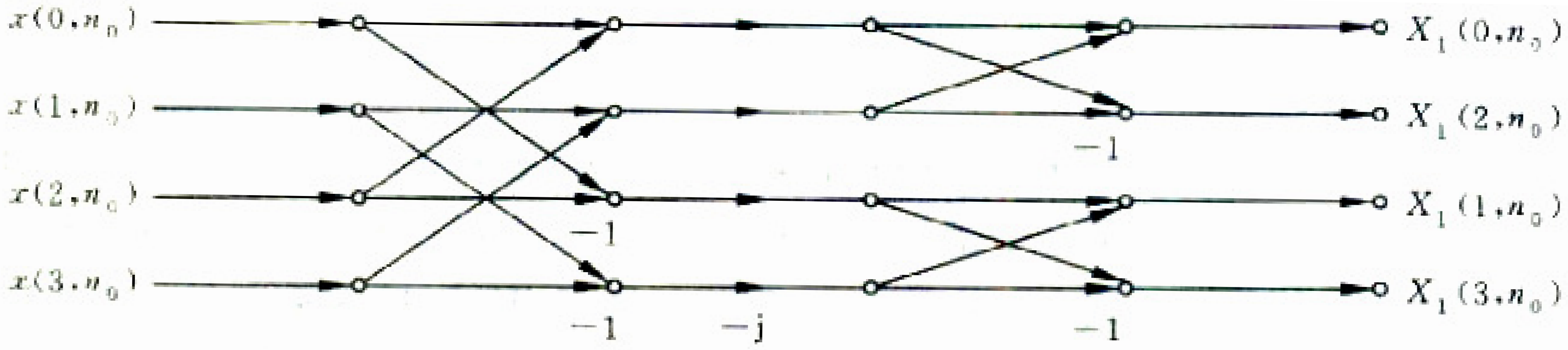
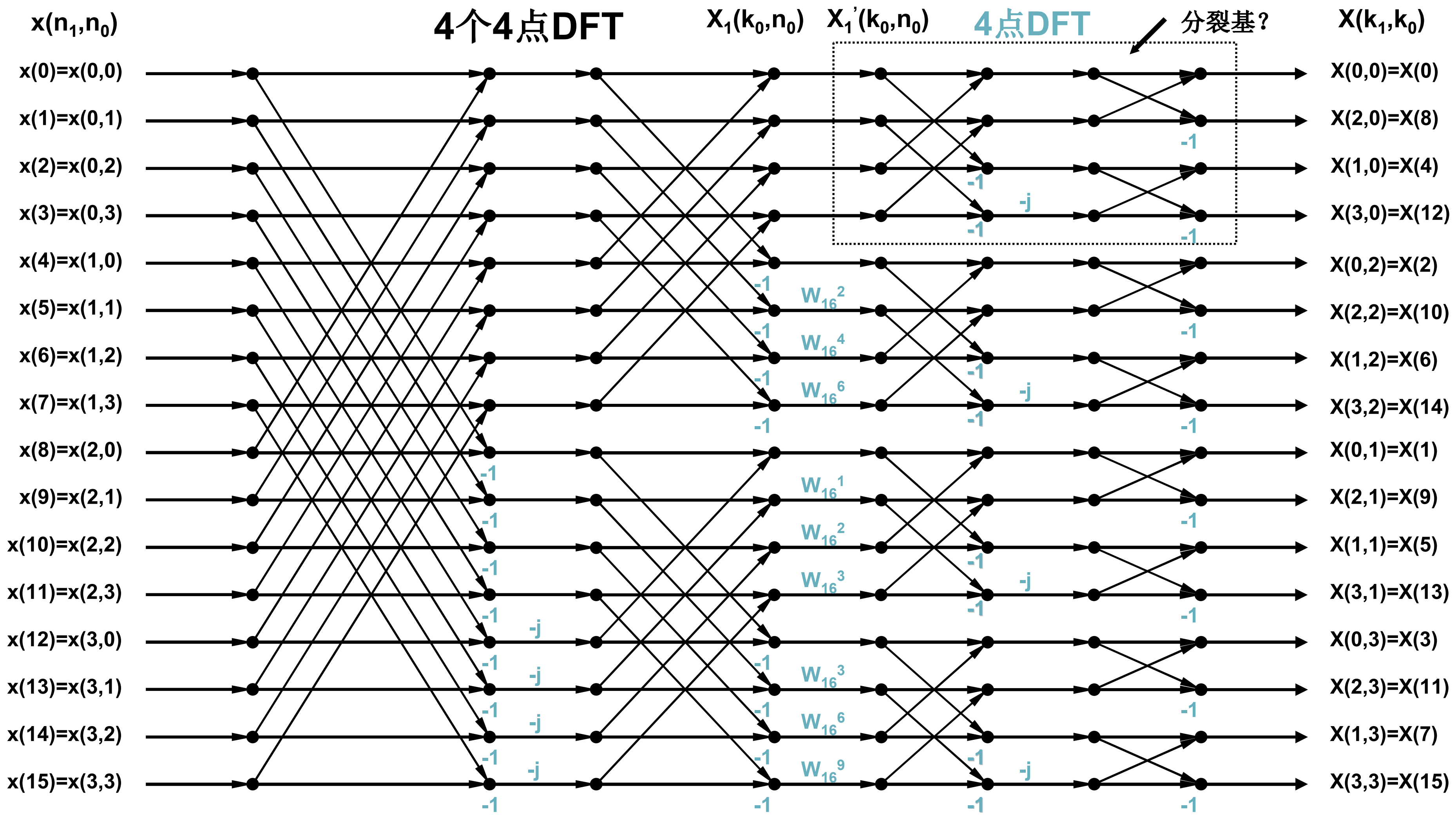


图 4-21　一个基-4 FFT 基本运算的信号流图



**16点基-4按频率抽取FFT算法流图

计算复杂度：基-4 FFT v.s. 基-2 FFT

- 若不考虑数据分解和组合的计算量，分解组合一次，基-2算法复数乘法由N^2降至$N^2/2$；基-4算法复数乘法则由N^2降至$N^2/4$；
- 当N较大时，基-4算法效率更高，但实现复杂度高；
- 对于长度为$2^L/4^L/8^L$...（L为正数）的序列，可通过L级2/4/8点的DFT计算整个序列的DFT，并谓之“基-2/-4/-8...FFT算法”；
- 基数越大，程序和硬件都越复杂，实际应用意义不大，而以基-2和基-4算法居多（后者比前者效率高，但实现复杂度也高）；

- **1984年，法国学者杜哈梅尔和霍尔曼将基-2和基-4算法揉和在一起，提出了所谓分裂基算法，它在计算效率和实现复杂度之间作了很好的折衷：其运算量较基-2算法有所减少，但运算流图却与基-2算法颇为接近，是一种实用、高效的FFT算法**

-> P.Duhamel & H. Hollmann, "Split-radix FFT algorithm," *Electronics Letters*, vol. 20, pp. 14-16, Jan. 1984

'SPLIT RADIX' FFT ALGORITHM

Indexing terms: Signal processing, Fast Fourier transforms

A new $N = 2^n$ fast Fourier transform algorithm is presented, which has fewer multiplications and additions than radix 2^n, $n = 1, 2, 3$ algorithms, has the same number of multiplications as the Rader–Brenner algorithm, but much fewer additions, and is numerically better conditioned, and is performed 'in place' by a repetitive use of a 'butterfly'-type structure.

Introduction: Since the early paper by Cooley and Tukey,[1] a lot of work has been done on the FFT algorithm, and this has resulted in classes of algorithms such as radix-2^m algorithms, the Winograd algorithm (WFTA), and prime factor algorithms (PFA).

Among these, the radix-2 and radix-4 algorithms have been used most for practical applications. This is due to their simple structure, with a constant geometry (butterfly type) and the possibility of performing them 'in place', even if they are more costly in terms of number of multiplications than WFTA and PFA.

Recently, some radix-2 algorithms have been proposed[2–4] which preserve more or less the advantages mentioned above, but require less multiplications than the usual radix-2^m algorithms. Unfortunately, these methods need 20–30% more additions, and seem to be numerically ill conditioned.

ELECTRONICS LETTERS 5th January 1984 Vol. 20 No. 1

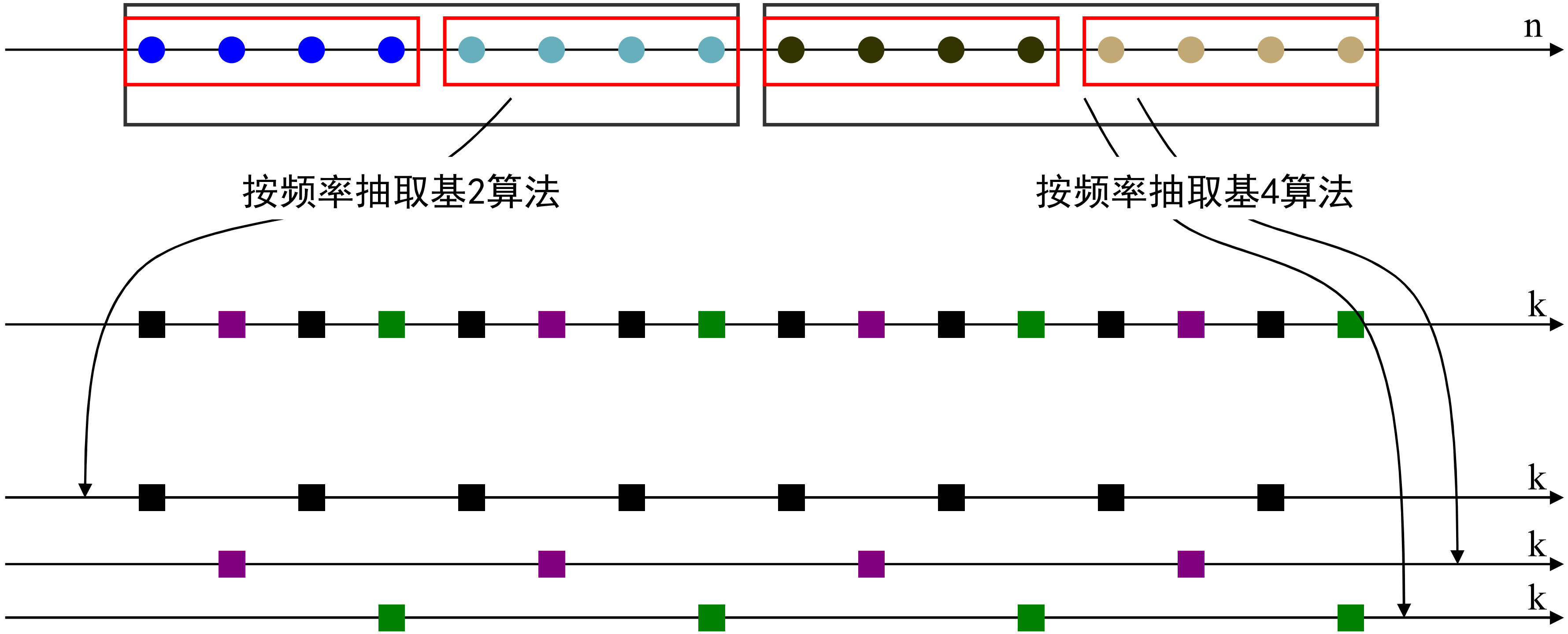
分裂基FFT算法原理：

- DFT偶数部分利用基2算法；奇数部分利用基4算法
- 采用时域分段，频域抽取的方法

偶数部分：时域分2段（n、n + N/2, n = 0 ~ N/2 – 1），频域抽取（2k, k = 0 ~ N/2 – 1）

奇数部分：时域分4段（n、n + N/4、n + 2N/4、n + 3N/4, n = 0 ~ N/4 – 1）

频域抽取2段（2(2k) + 1 = 4k + 1、2(2k + 1) + 1 = 4k + 3）



合并$X(4k)$与$X(4k+2)$:

$$
\begin{cases}
X(2k)=\sum_{n=0}^{\frac{N}{2}-1}[x(n)+x(n+\frac{N}{2})]W_N^{2kn}, \quad 0\le k\le \frac{N}{2}-1 \\
X(4k+1)=\sum_{n=0}^{\frac{N}{4}-1}\left\{\left[\left(x(n)-x(n+\frac{N}{2})\right)-j\left(x(n+\frac{N}{4})-x(n+\frac{3N}{4})\right)\right]W_N^{n}\right\}W_N^{4kn} \\
X(4k+3)=\sum_{n=0}^{\frac{N}{4}-1}\left\{\left[\left(x(n)-x(n+\frac{N}{2})\right)+j\left(x(n+\frac{N}{4})-x(n+\frac{3N}{4})\right)\right]W_N^{3n}\right\}W_N^{4kn}
\end{cases}
$$

$$0\le k\le \frac{N}{4}-1$$

令

$$
\begin{cases}
x_2(n) = \overbrace{x(n)+x(n+\frac{N}{2})}^{2次}, \quad 0 \le n \le \frac{N}{2}-1 \\
x_4^1(n) = \left[\left(x(n)-x(n+\frac{N}{2})\right) \underbrace{-j\left(x(n+\frac{N}{4})\right.}_{1次} \left.-x(n+\frac{3N}{4})\right)\right] W_N^n \\
x_4^2(n) = \left[\left(\underbrace{x(n)-x(n+\frac{N}{2})}_{1次}\right) \underbrace{+j\left(x(n+\frac{N}{4})\right.}_{1次} \underbrace{\left.-x(n+\frac{3N}{4})\right)}_{1次}\right] W_N^{3n}
\end{cases}
$$

$$0 \le n \le \frac{N}{4}-1$$

L形蝶形运算：×— 2次

+— 6次

则：

$$
\begin{cases}
X(2k) = \sum_{n=0}^{\frac{N}{2}-1} x_2(n) W_N^{2kn} = DFT[x_2(n)] \\
X(4k+1) = \sum_{n=0}^{\frac{N}{4}-1} x_4^1(n) W_N^{4kn} = DFT[x_4^1(n)] \\
X(4k+3) = \sum_{n=0}^{\frac{N}{4}-1} x_4^2(n) W_N^{4kn} = DFT[x_4^2(n)]
\end{cases}
$$

分裂基 FFT （SRFFT）：L 形蝶形计算

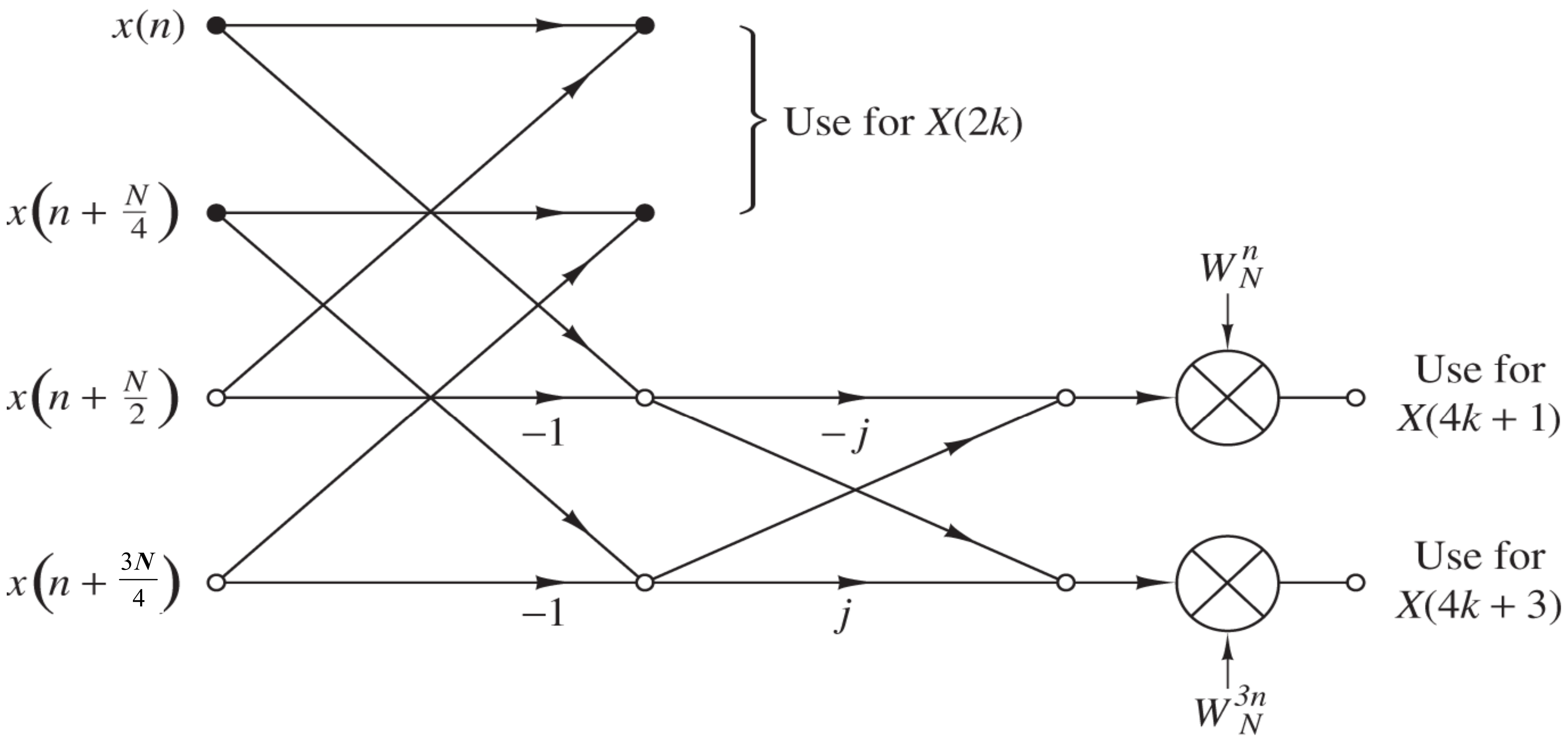


Figure 8.1.16 Butterfly for SRFFT algorithm.

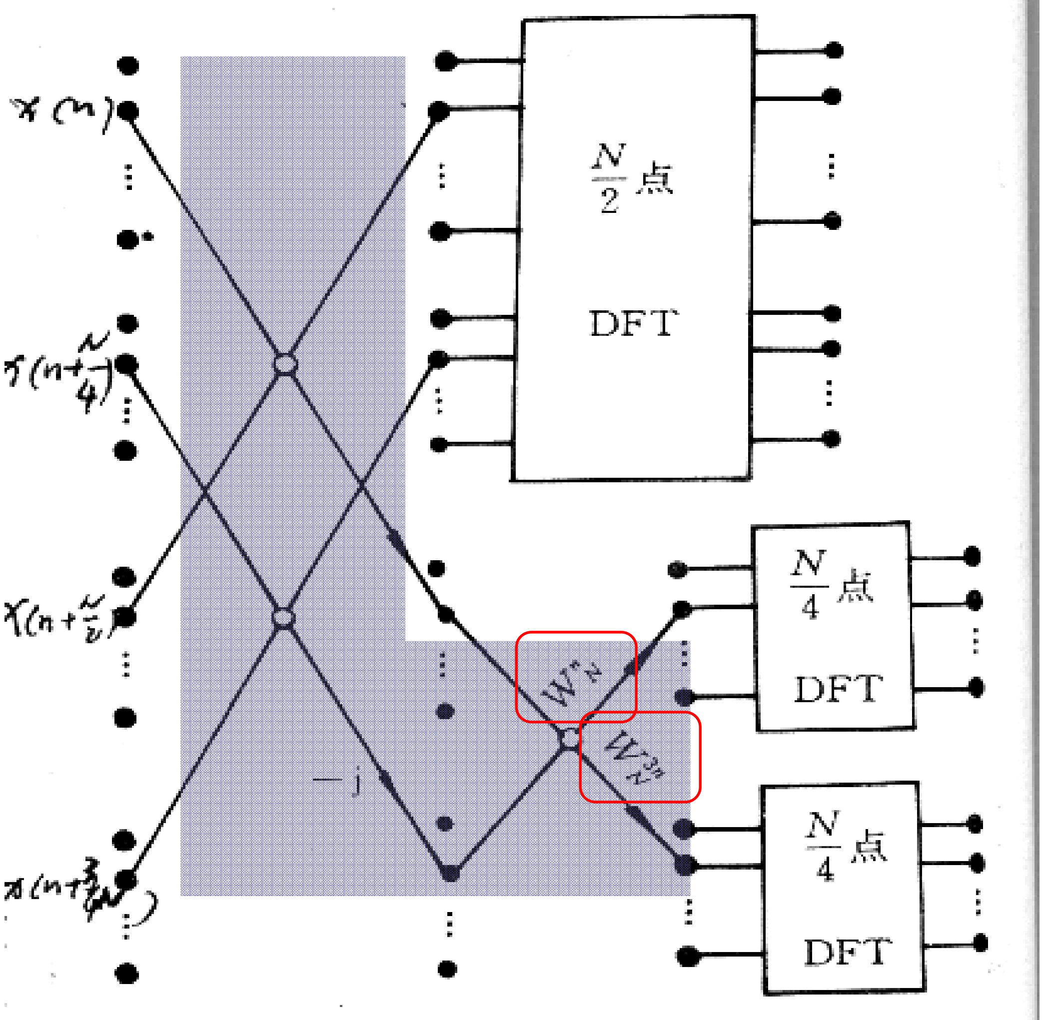


图 4—21　分裂基算法第一次分解 L 形流图

$$X(2k) = \sum_{n=0}^{\frac{N}{2}-1} x_2(n) W_N^{2kn}$$
$$= DFT[x_2(n)]$$

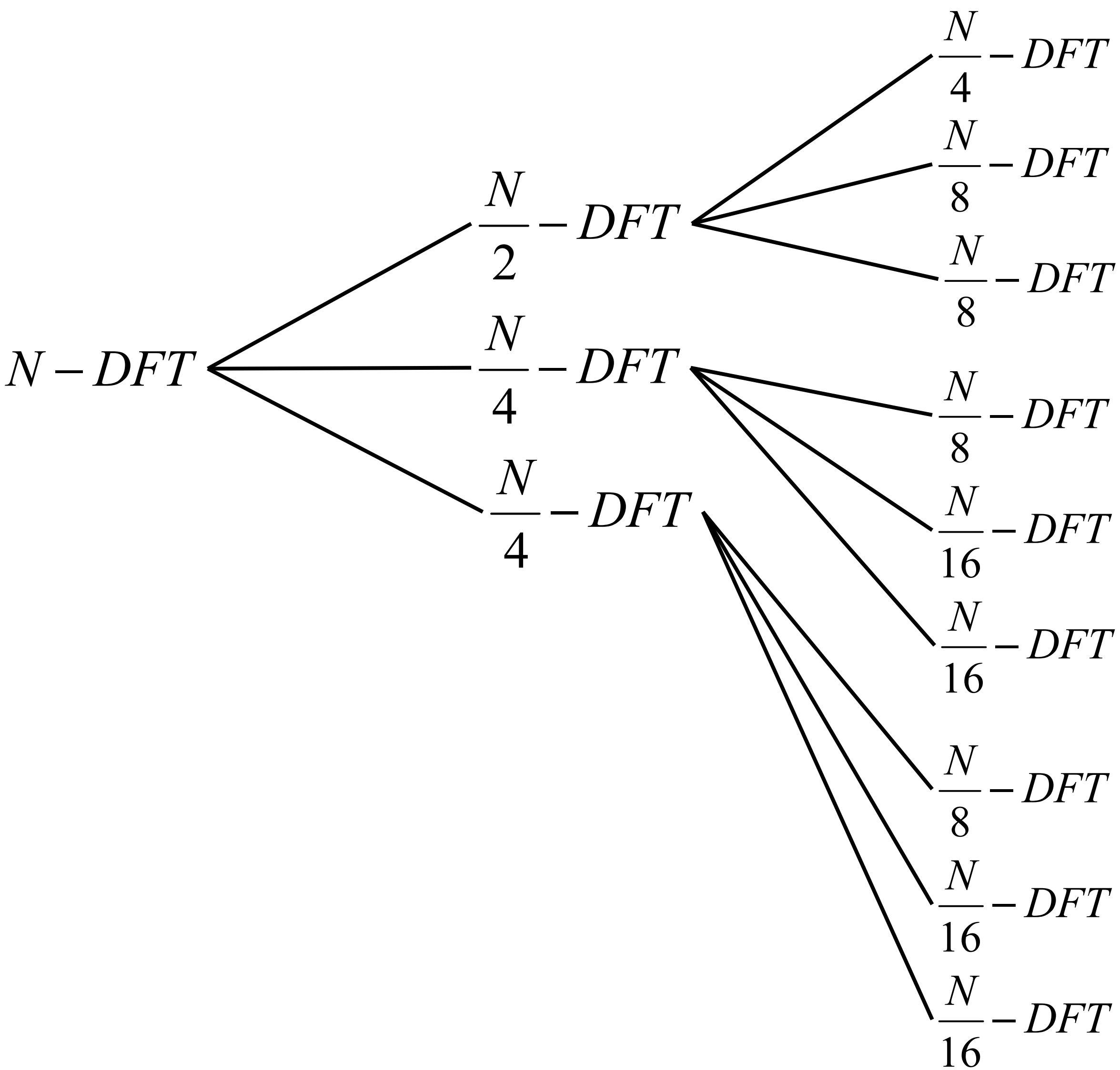
$$X(4k+1) = \sum_{n=0}^{\frac{N}{4}-1} x_4^1(n) W_N^{4kn}$$
$$= DFT[x_4^1(n)]$$

$$X(4k+3) = \sum_{n=0}^{\frac{N}{4}-1} x_4^2(n) W_N^{4kn}$$
$$= DFT[x_4^2(n)]$$

L形蝶形(流图)

与基4比较W_N

可见：



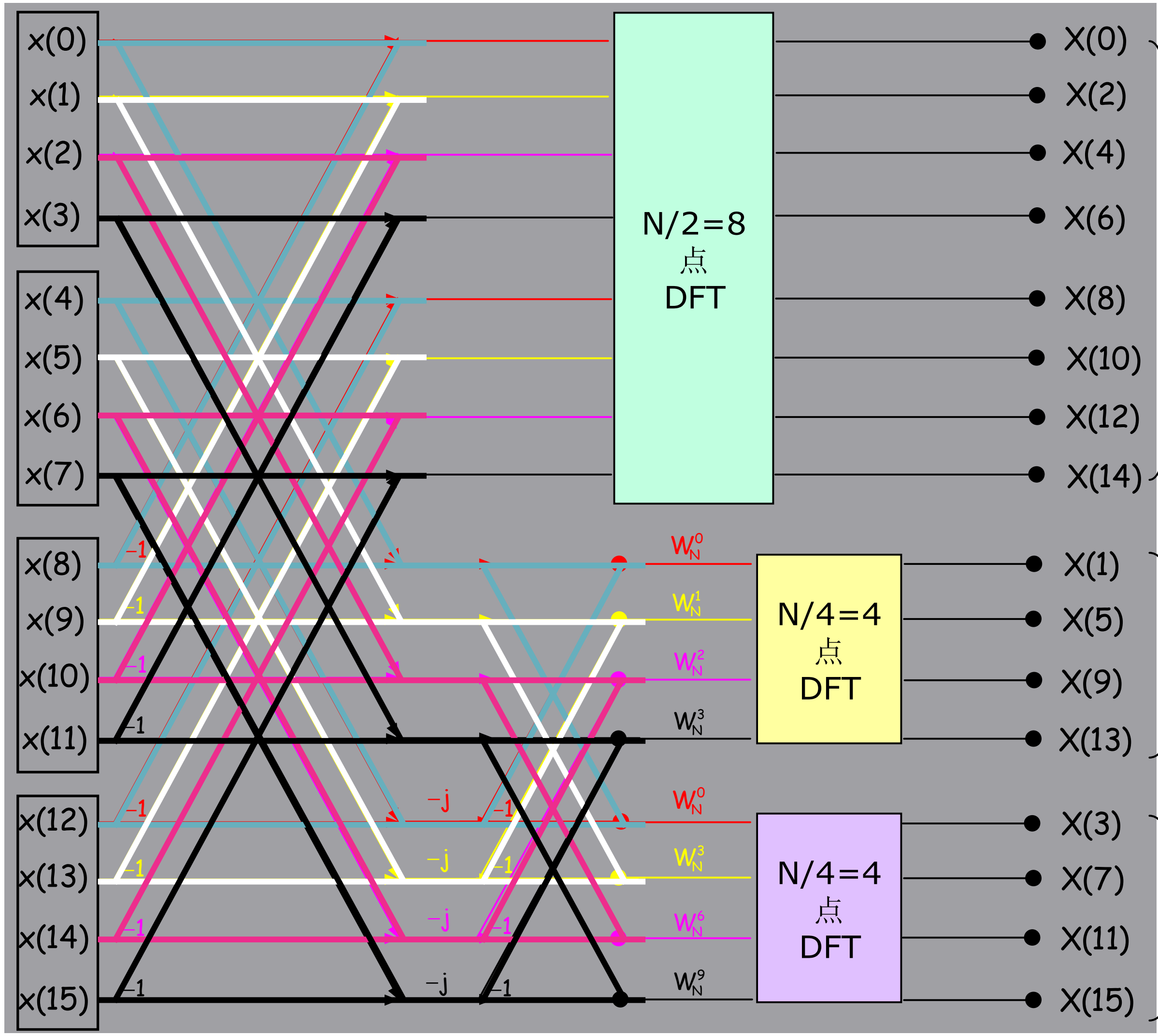
$N-DFT$
$\frac{N}{2}-DFT$
$\frac{N}{4}-DFT$
$\frac{N}{4}-DFT$
$\frac{N}{4}-DFT$
$\frac{N}{8}-DFT$
$\frac{N}{8}-DFT$
$\frac{N}{8}-DFT$
$\frac{N}{16}-DFT$
$\frac{N}{16}-DFT$
$\frac{N}{8}-DFT$
$\frac{N}{16}-DFT$
$\frac{N}{16}-DFT$

例：N=16， 分裂基FFT

分解1： $x_1(n) \to x_2(n) \qquad 0 \le n \le 7 \qquad (\frac{N}{2}-1)$

$x_4^1(n) \qquad 0 \le n \le 3 \qquad (\frac{N}{4}-1)$

$x_4^2(n) \qquad 0 \le n \le 3 \qquad (\frac{N}{4}-1)$



长偶

$$X(2k) = DFT[x(n) + x(n + N/2)]$$

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

长奇之偶

$$X(4k+1) = DFT\{[x(n) - x(n + N/2)] - j[x(n + N/4) - x(n + 3N/4)]\}W_N^n$$

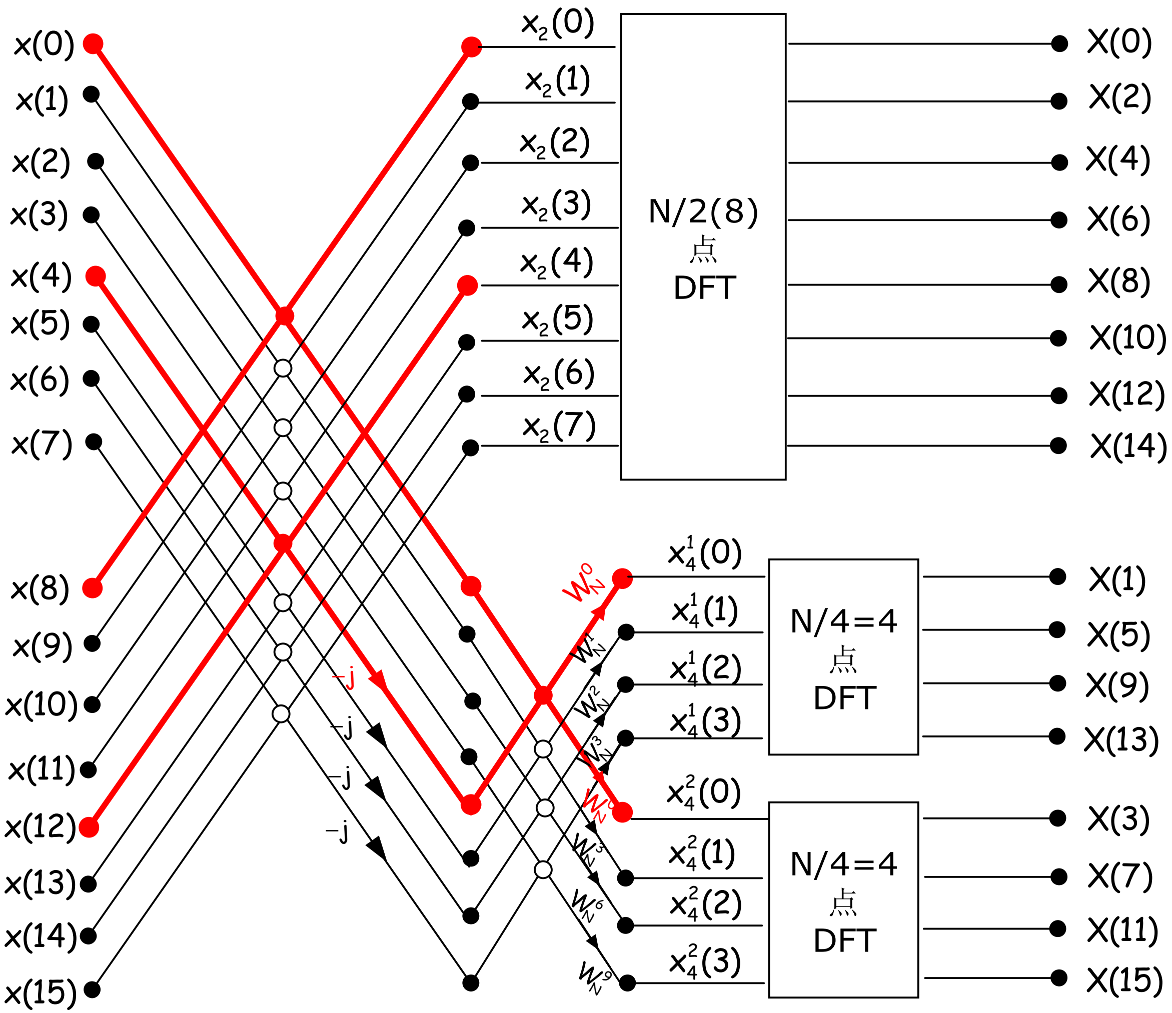
$$k = 0,1,2,\ldots,N/4-1$$

长奇之奇

$$X(4k+3) = DFT\{[x(n) - x(n + N/2)] + j[x(n + N/4) - x(n + 3N/4)]\}W_N^{3n}$$

$$k = 0,1,2,\ldots,N/4-1$$

时域分段组合四入四出“L”蝶形（计有N/4个）



alternatively!

分解2： $x_2(n) \to y_2(n) \quad 0 \le n \le 3 \quad (\frac{N}{4}-1) = \frac{N/2}{2}-1$

$y_4^1(n) \quad 0 \le n \le 1 \quad (\frac{N}{8}-1) = \frac{N/2}{4}-1$

$y_4^2(n) \quad 0 \le n \le 1 \quad (\frac{N}{8}-1) = \frac{N/2}{4}-1$

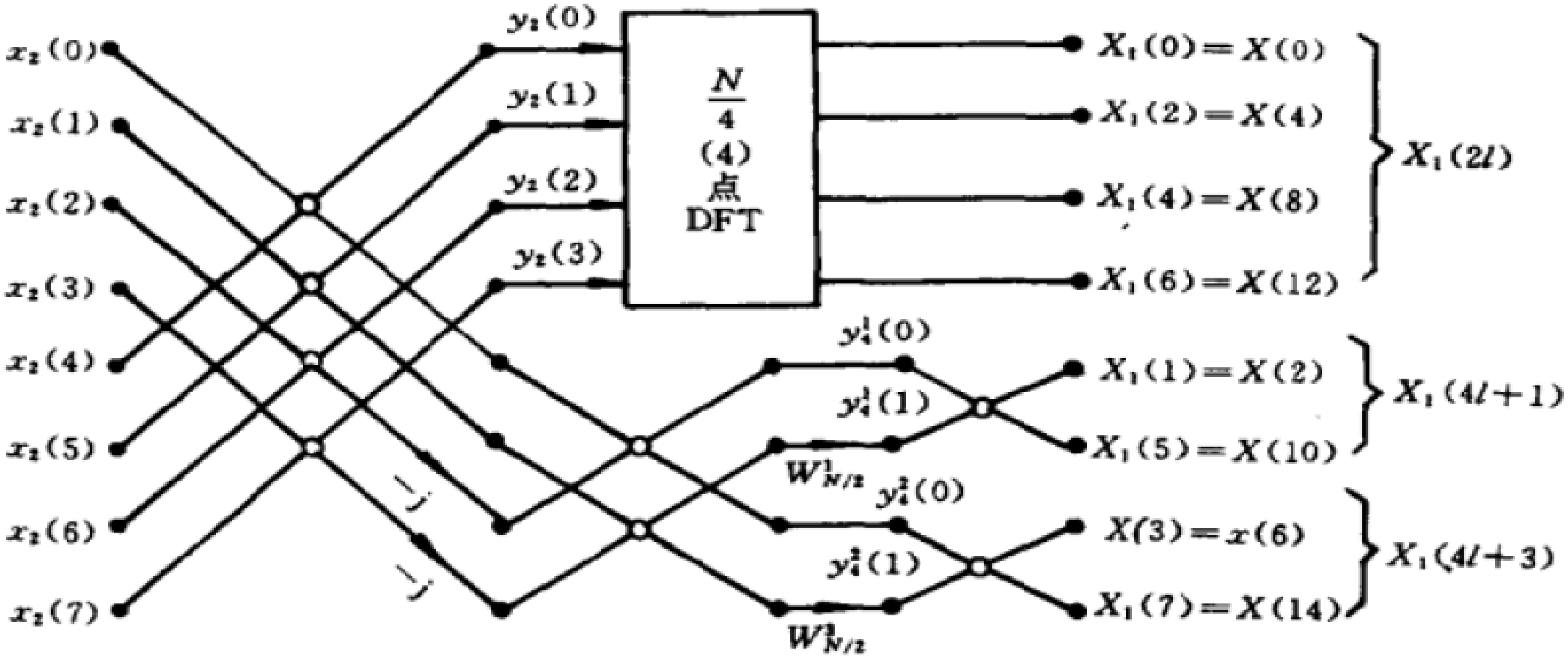


图 4—23　图 4—22 中的 $N/2$ 点 DFT 分解的 L 形流图

分解3：

$$y_2(n) \to z_2(n) \qquad 0 \le n \le 1$$

$$z_4^1(n) \qquad n = 0$$

$$z_4^2(n) \qquad n = 0$$

→ 4点分裂基L形运算流图

$$x_4^1(n) \to \cdots\cdots$$

$$x_4^2(n) \to \cdots\cdots$$

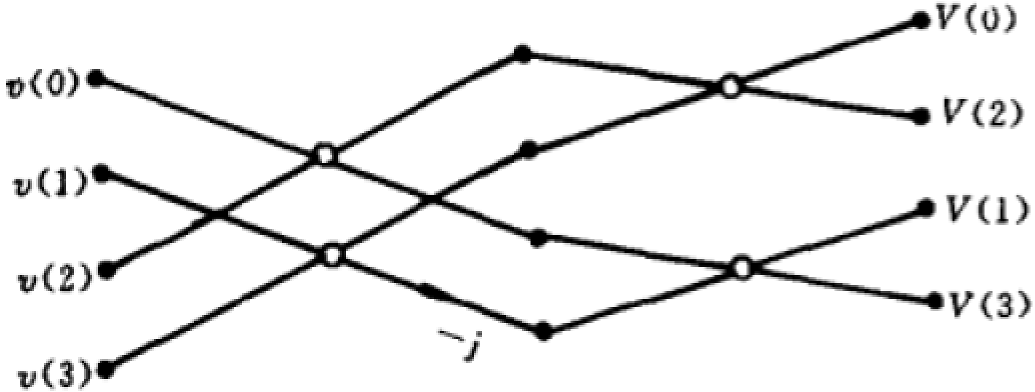


图 4－24　4 点分裂基 L 形运算流图

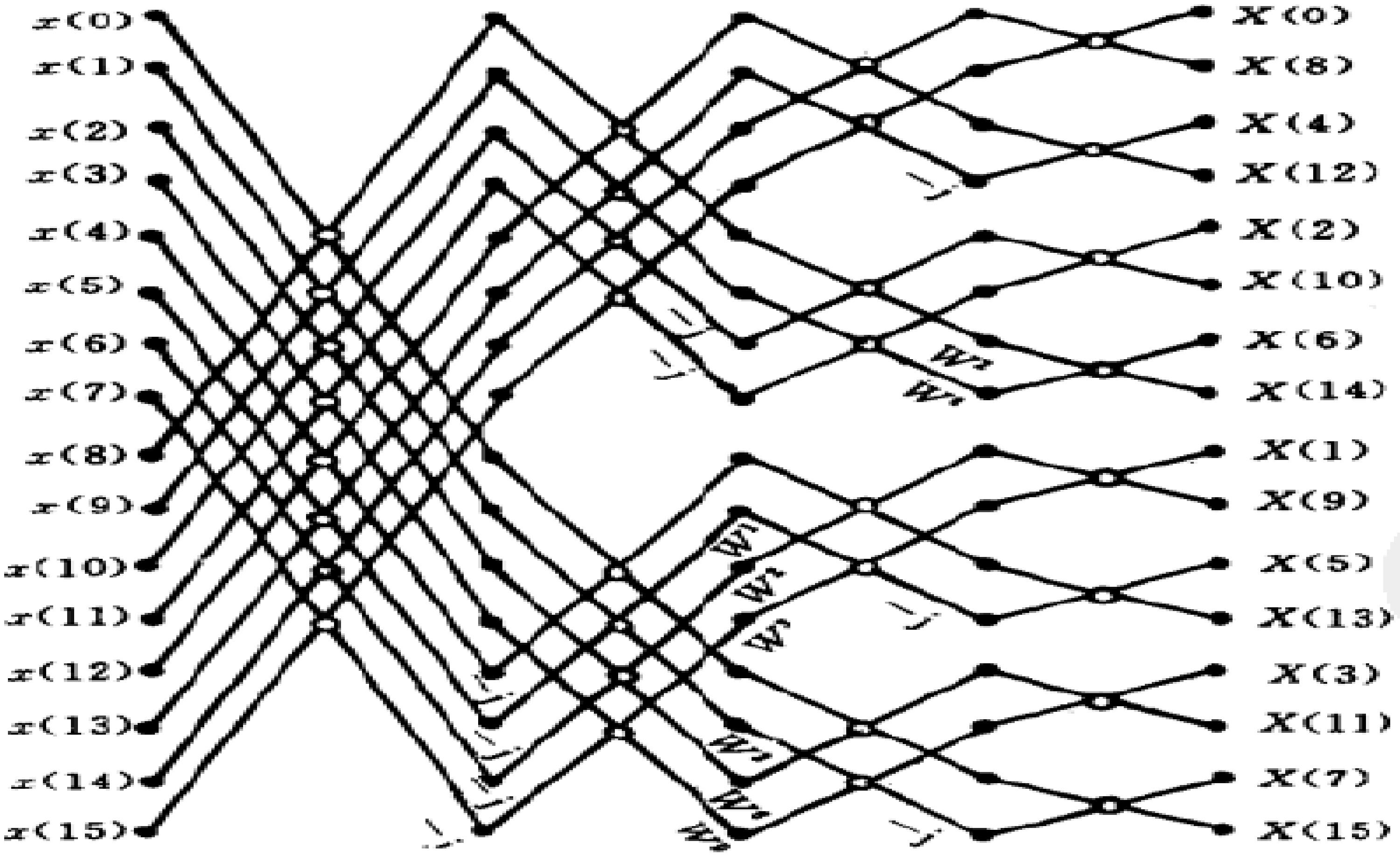
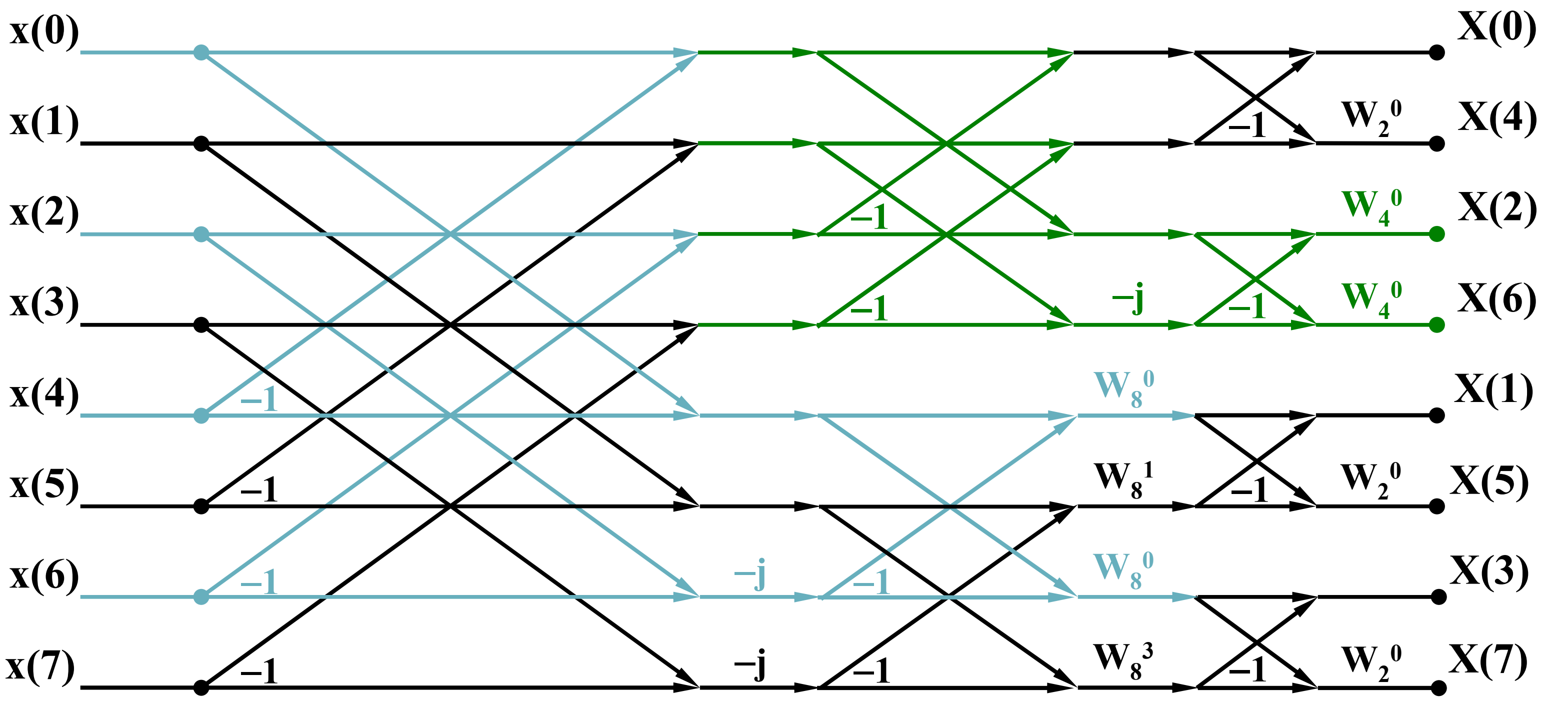
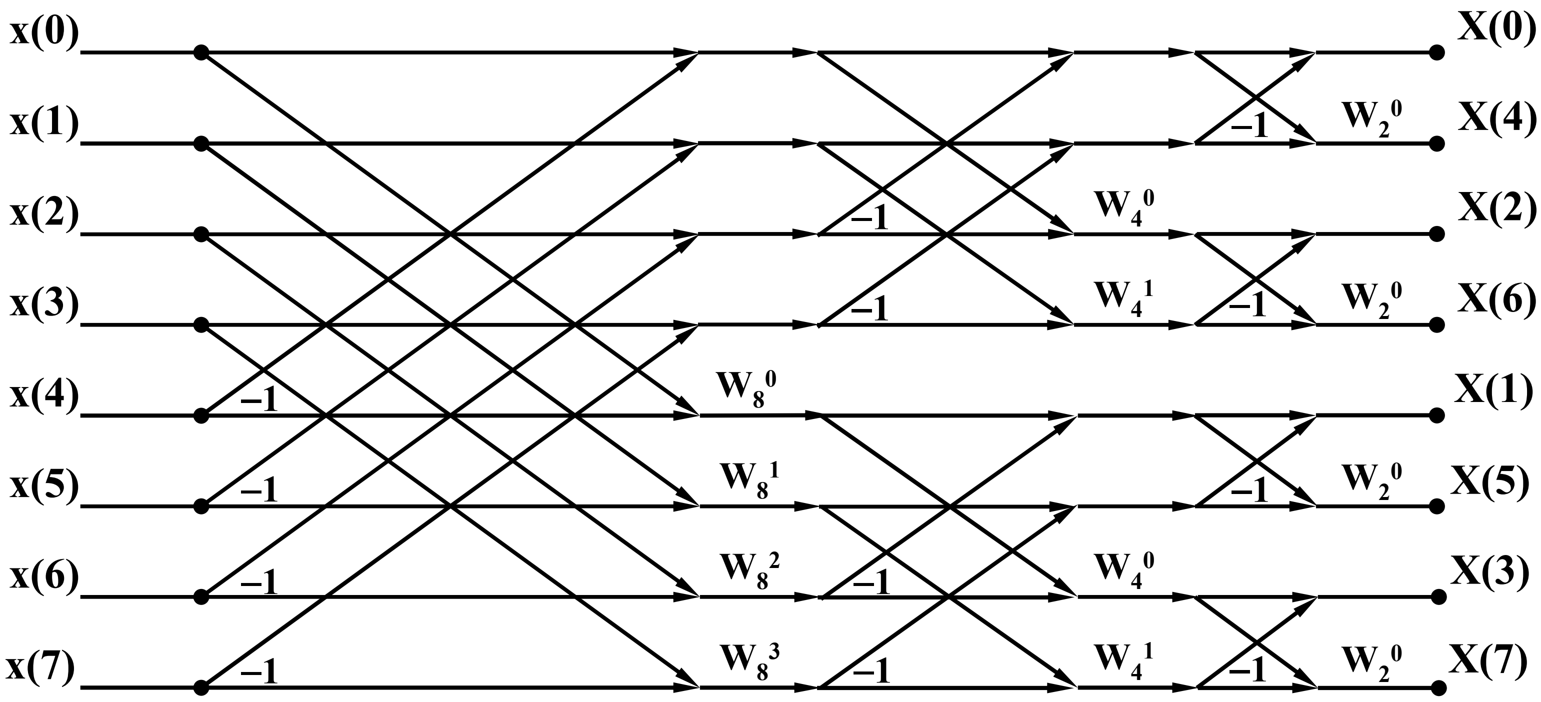


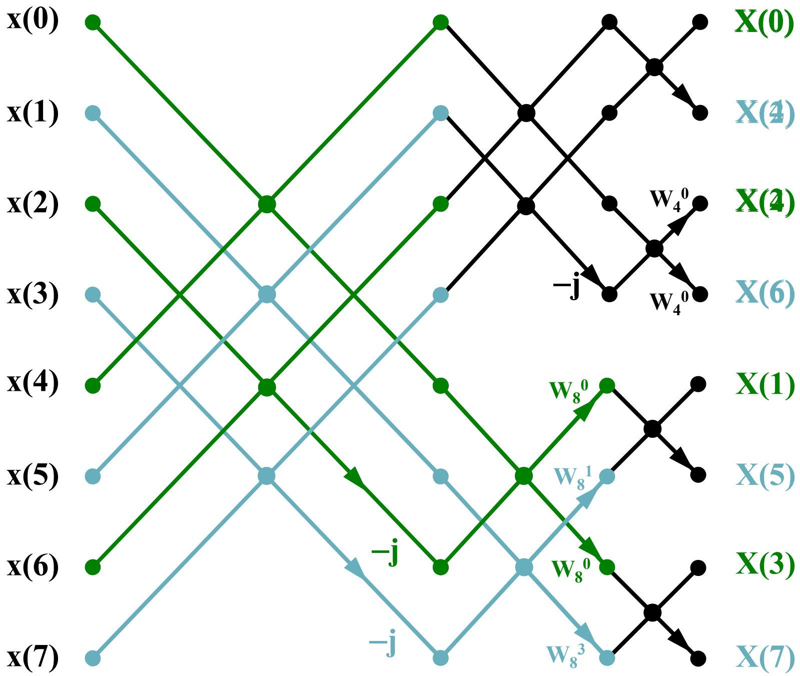
图 4—25　16 点分裂基按频率抽取的 FFT 算法流图
(图中各支路上箭头均已略去)



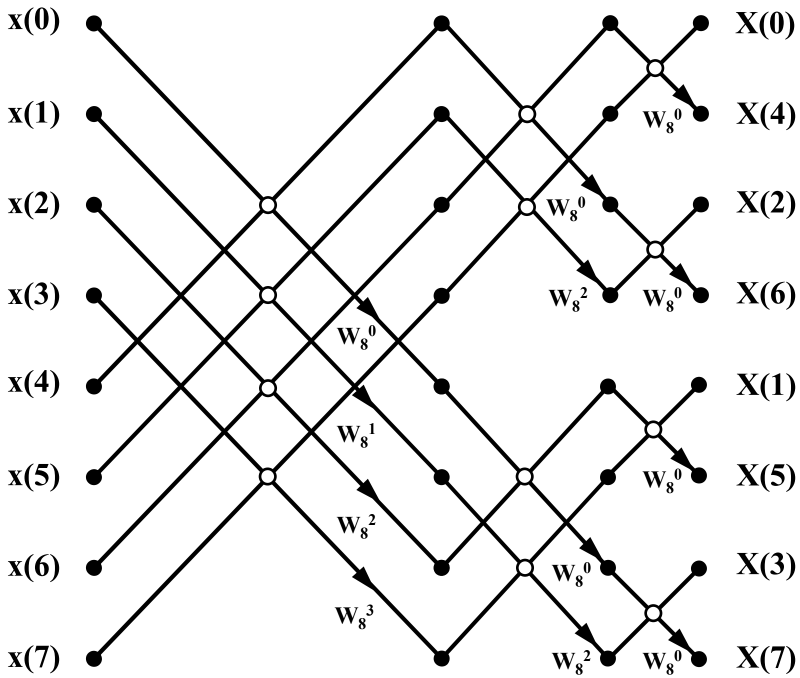
8点分裂基FFT实现流图

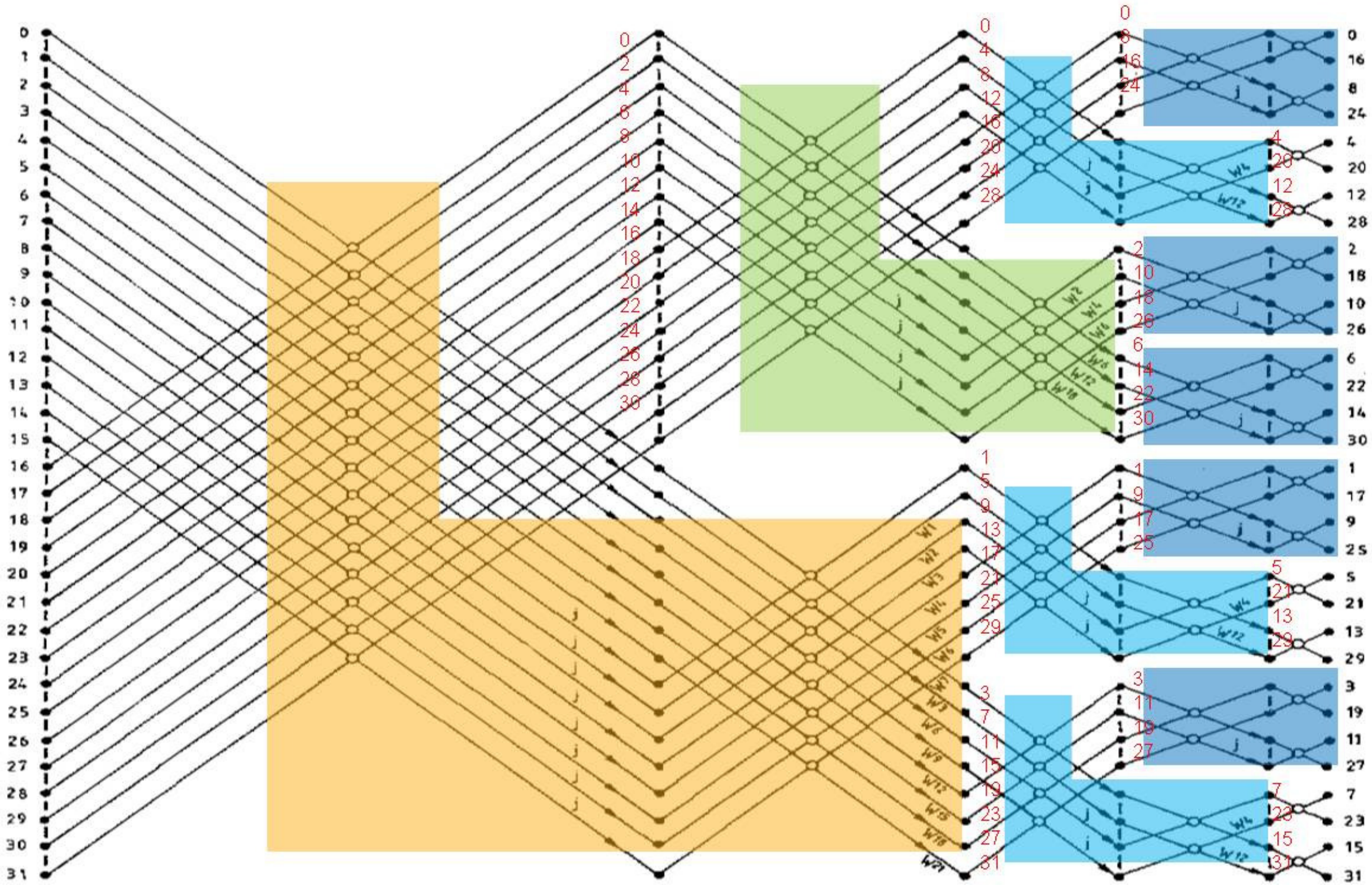


8点按频率抽取基2-FFT实现流图



alternatively!





N=32 蝶形个数：8 4 6 5

N=16 分裂基 按时间抽取 FFT算法流图

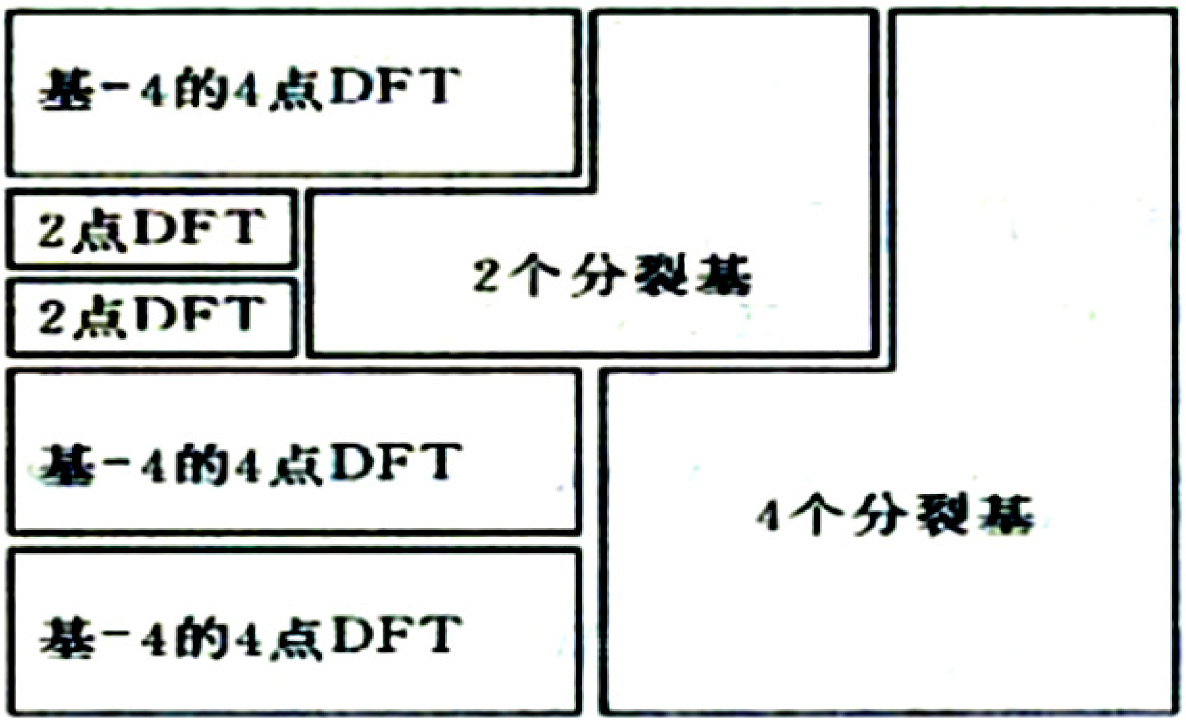


图 4-25 $N=2^4=16$ 点的分裂基 FFT 的示意图

N=16 分裂基 按时间抽取 FFT算法流图

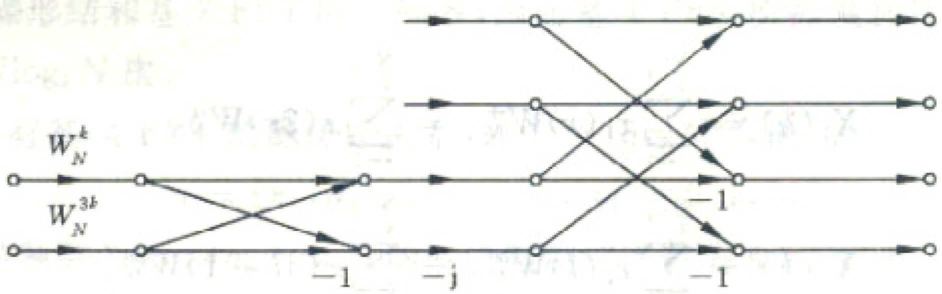


图 4-24 分裂基 FFT 算法的一个基本蝶形运算

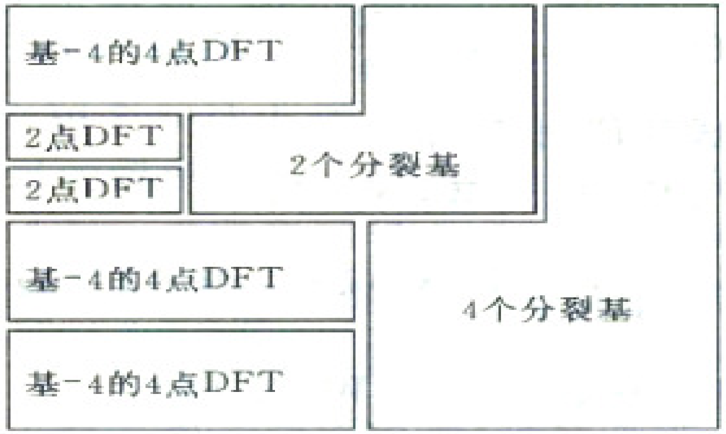


图 4-25 $N=2^4=16$ 点的分裂基 FFT 的示意图

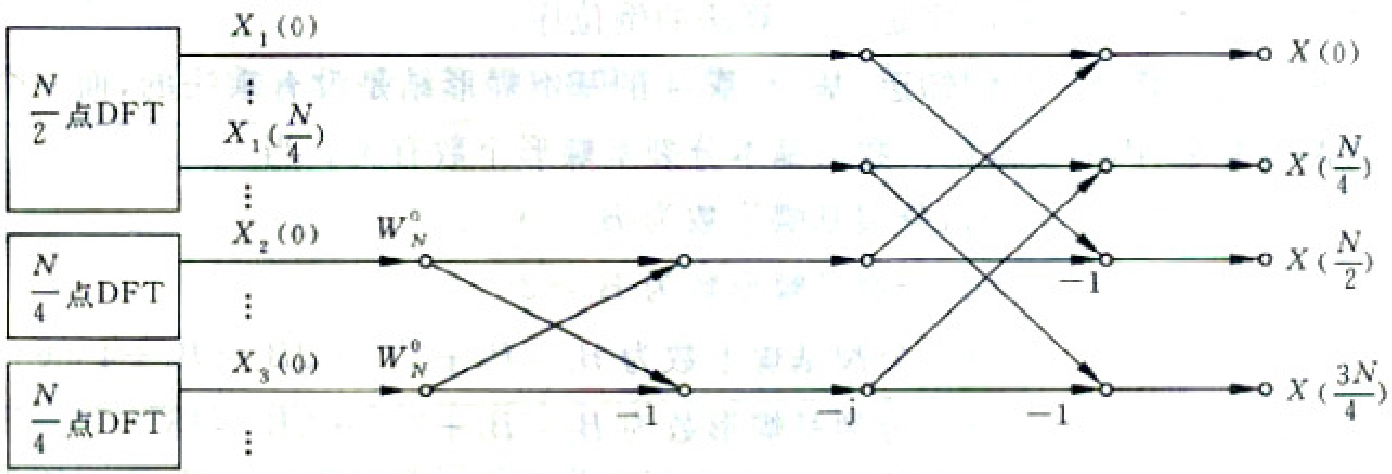


图 4-23 分裂基 FFT 算法(时间抽选)的第一级流图

三、运算量分析

L形分解：共M-1级　　$N=2^M$

每级L形碟形个数：

$$l_1=\frac{N}{4}$$

$$l_j=\frac{N}{4}-\frac{l_{j-1}}{2},\qquad j=2,3,\dots,M-1$$

每个L形碟形：×— 2次

总的复数乘法次数：

$$C_M=2\times\sum_{j=1}^{M-1}l_j=\frac{1}{3}N\log_2^N-\frac{1}{9}N+(-1)^M\frac{2}{9}$$

相比 $\frac{N}{2}\log_2^N$，下降33% $=(\frac{1}{2}-\frac{1}{3})\Big/\frac{1}{2}\times100\%$

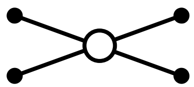
$+\,—N\log_2^N$ 相同（∵  个数相同）

TABLE 8.2 Number of Nontrivial Real Multiplications and Additions to Compute an N-point Complex DFT

	Real Multiplications				Real Additions			
N	Radix 2	Radix 4	Radix 8	Split Radix	Radix 2	Radix 4	Radix 8	Split Radix
16	24	20		20	152	148		148
32	88			68	408			388
64	264	208	204	196	1,032	976	972	964
128	712			516	2,504			2,308
256	1,800	1,392		1,284	5,896	5,488		5,380
512	4,360		3,204	3,076	13,566		12,420	12,292
1,024	10,248	7,856		7,172	30,728	28,336		27,652

Source: Extracted from Duhamel (1986).

基-2、基-4、分裂基按频率抽取 FFT算法流图X(k)输出序号是否相同？

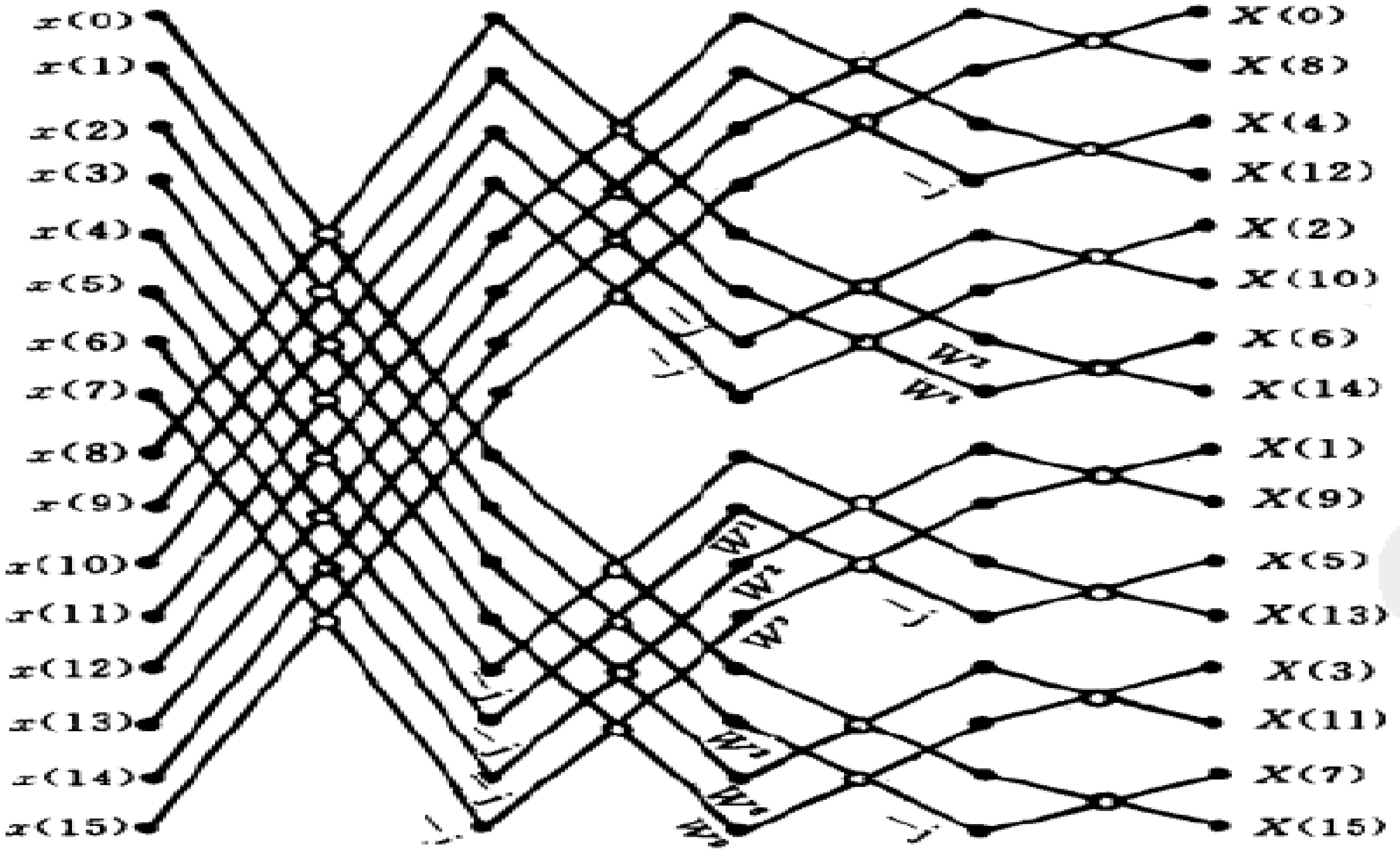


图 4—25　16 点分裂基按频率抽取的 FFT 算法流图
（图中各支路上箭头均已略去）

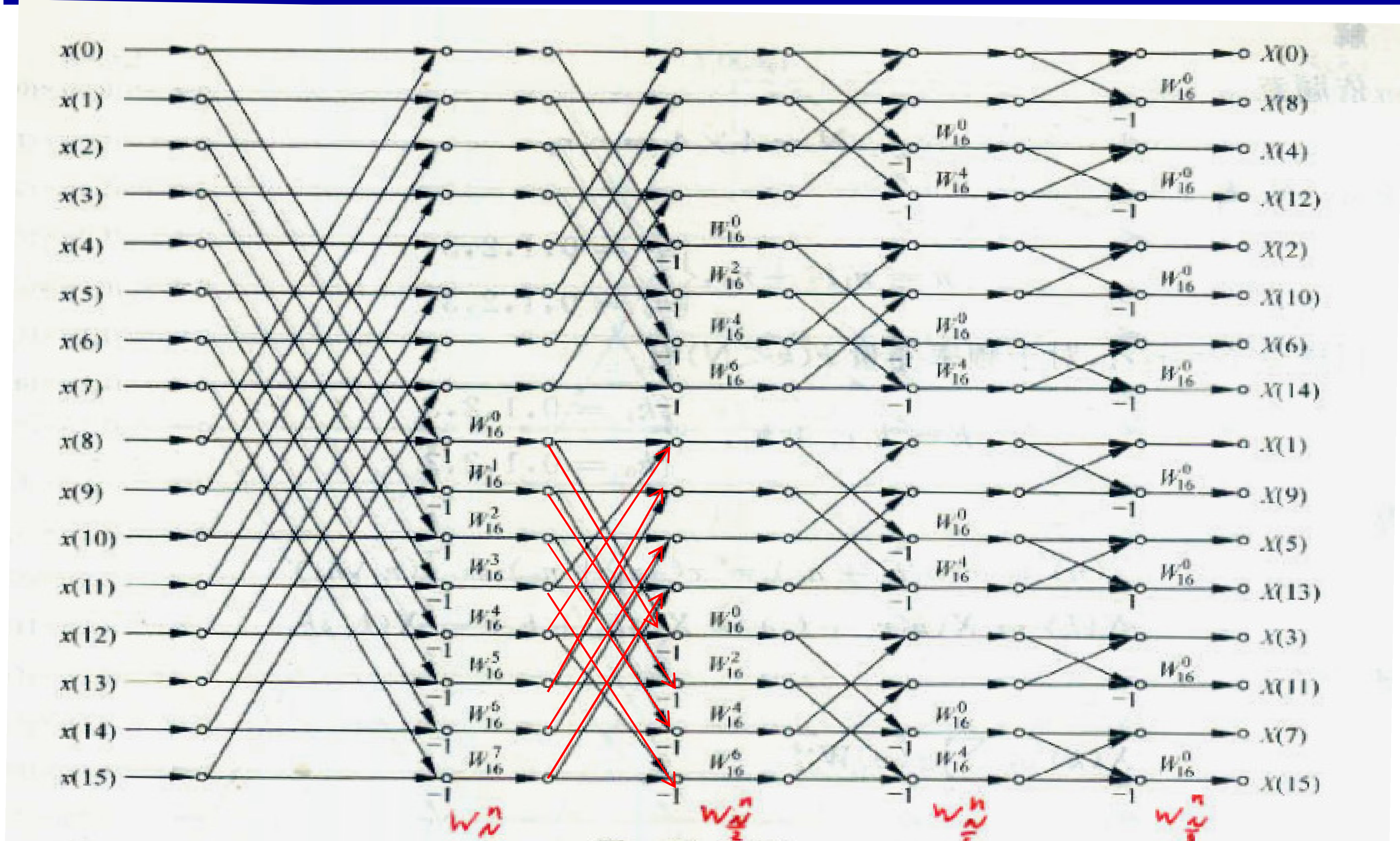


图 P4-3(b)

N=16 基-2 按频率抽取FFT流图

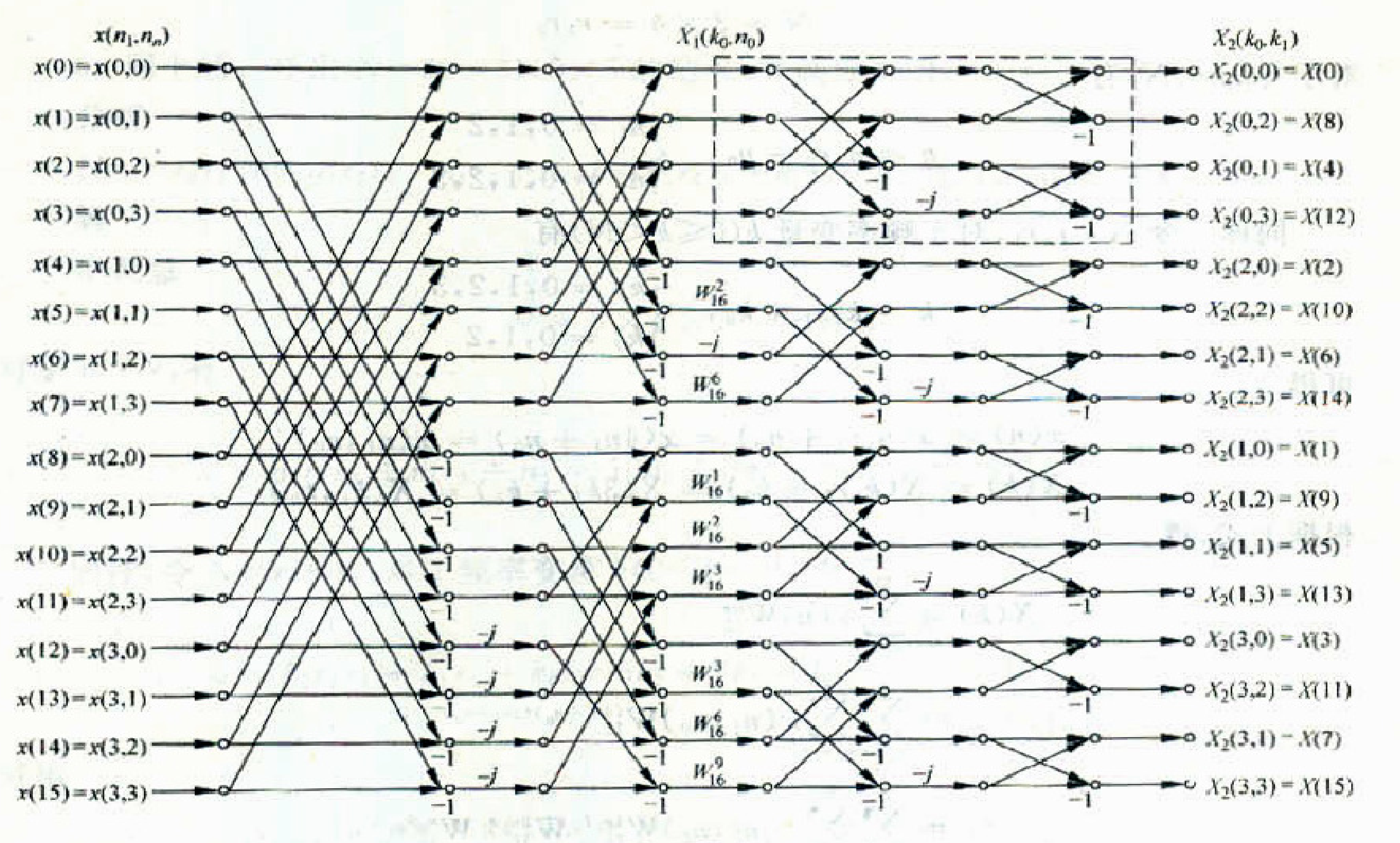


图 P4-4

N=16 基-4 按频率抽取FFT流图

基-2、基-4、分裂基按时间抽取 FFT算法流图x(n)输入序号是否相同?

N=16 分裂基 按时间抽取 FFT算法流图

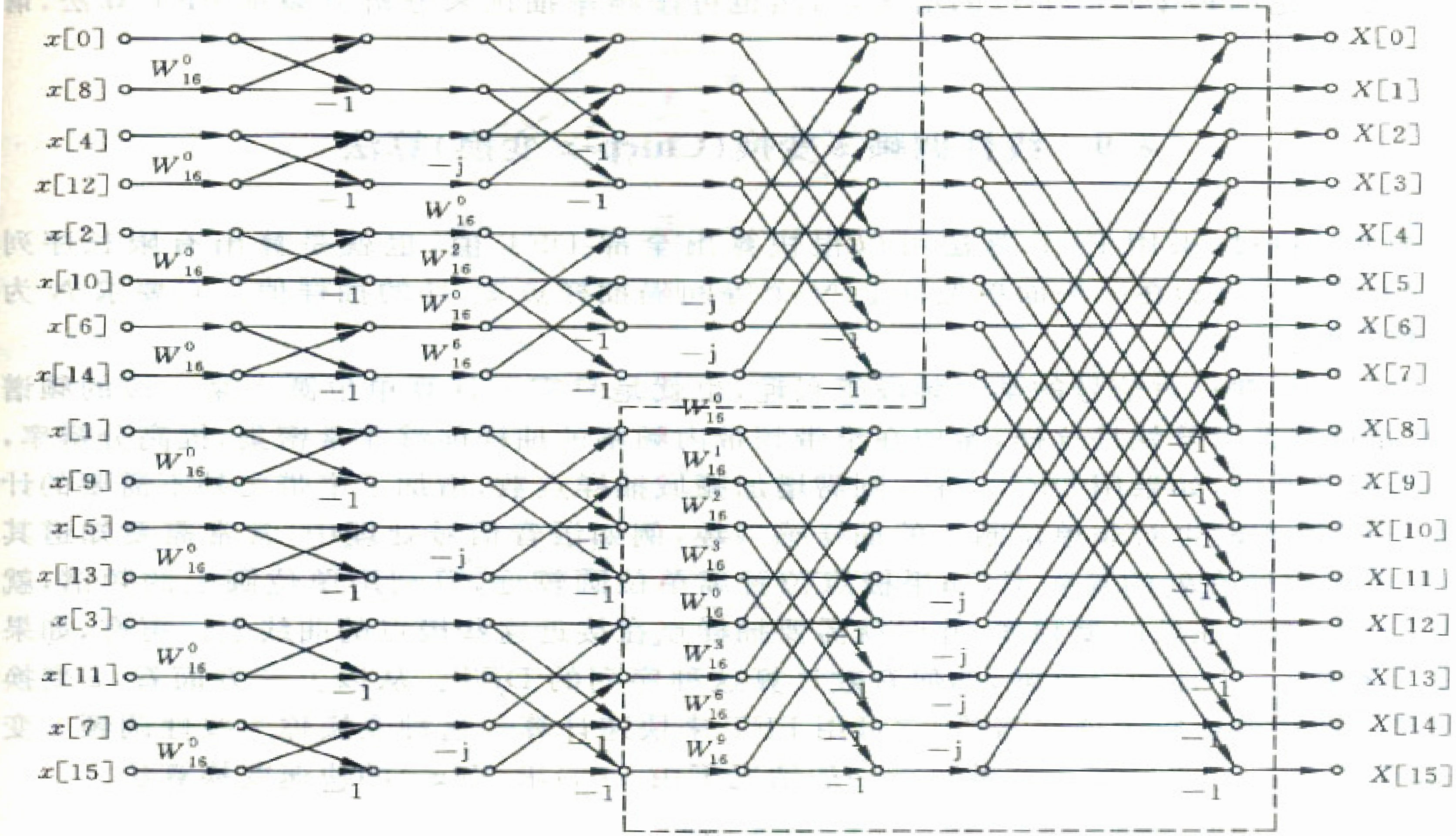


图 4-26　$N=2^4=16$ 分裂基 FFT 算法(按时间抽选)的流图

(输入二进制倒位序,输出正常顺序)

注：上图只用虚线框表示了一级的倒 L 结构

N=16 基-2 按时间抽取FFT流图

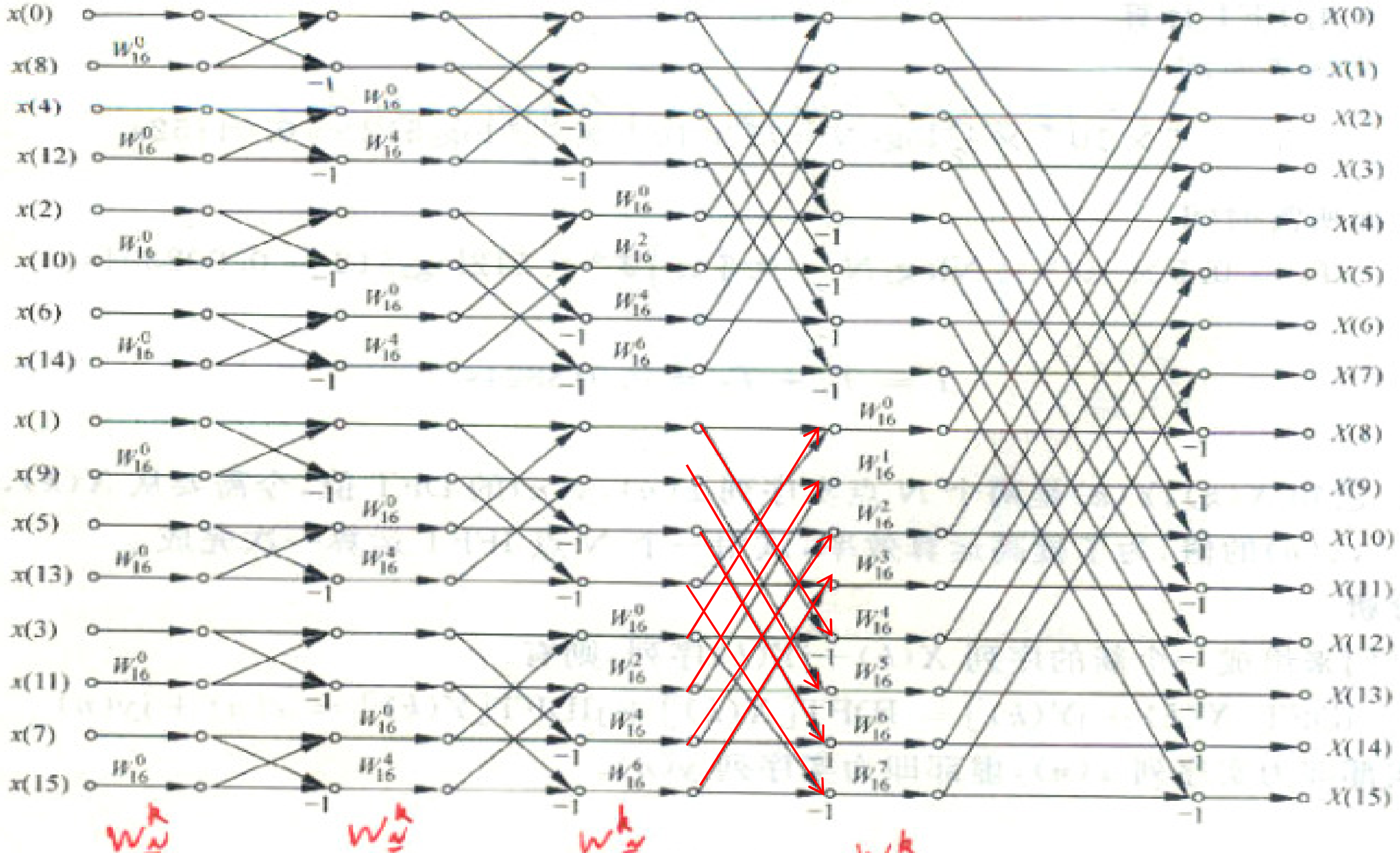


图 P4-3(a)