武汉大学 信号处理研究室

《数字信号处理》

Digital Signal Processing

余磊　副教授

武汉大学电子信息学院
Email: ly.wd@whu.edu.cn

本科生课程, 2018 秋

Outline

DFT 变换

离散傅里叶变换 (DFT)

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

DFT 变换的矩阵形式

- Rewrite DFT $X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$ as a matrix multiply:

$$\underbrace{\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ \vdots \\ X[N-1] \end{bmatrix}}_{\mathbf{X}} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N^1 & W_N^2 & \cdots & W_N^{(N-1)} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{(N-1)} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)^2} \end{bmatrix}}_{\mathbf{D}_N} \underbrace{\begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-1] \end{bmatrix}}_{\mathbf{x}}$$

$$\mathbf{X} = \mathbf{D}_N \cdot \mathbf{x}$$

DFT 计算复杂度分析

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

- $x[n]W_N^{kn} \Rightarrow$ 一次复数乘法 $\xrightarrow{\text{N 点 DFT}}$ N 次复数乘法
- $\sum_{n=0}^{N-1} \Rightarrow$ N-1 次复数加法

  - cpx mult: $(a+jb)(c+jd) = ac - bd + j(ad + bc)$
    = 4 real mults + 2 real adds
  - cpx add = 2 real adds

计算 $X[k]$ 的复杂度为：$4N$ 次实数乘法 $+4N-2$ 次实数加法
$\Rightarrow$ N 点 DFT（$k=1...,N$）需要$4N^2$ 次实数乘法 $+(4N^2-2N)$ 次实数加法

History of FFT

- 1807 年，傅里叶发明了傅里叶变换

History of FFT

- 1807 年，傅里叶发明了傅里叶变换
- 1958 年，戈泽尔发明了戈泽尔算法（速度提升 ≈2 倍）

History of FFT

- 1807 年，傅里叶发明了傅里叶变换
- 1958 年，戈泽尔发明了戈泽尔算法（速度提升 $\approx$2 倍）
- 1965 年，Cooley 和 Tukey 『发明』了快速傅里叶变换

$$O(N^2) \rightarrow O(N \log N)$$

History of FFT

- 1807 年，傅里叶发明了傅里叶变换
- 1958 年，戈泽尔发明了戈泽尔算法（速度提升 ≈2 倍）
- 1965 年，Cooley 和 Tukey『发明』了快速傅里叶变换

$$O(N^2) \rightarrow O(N \log N)$$

- **1805 年，Gauss(高斯，28 岁) 在计算小行星的轨道时使用了该算法**

History of FFT

- 1807 年，傅里叶发明了傅里叶变换
- 1958 年，戈泽尔发明了戈泽尔算法（速度提升 ≈2 倍）
- 1965 年，Cooley 和 Tukey『发明』了快速傅里叶变换

$$O(N^2) \rightarrow O(N \log N)$$

- **1805 年，Gauss(高斯，28 岁) 在计算小行星的轨道时使用了该算法**
- **1942 年，Danielson 和 Lanczos 也描述了同样的算法，但是那个时候没有被人们重视**

Outline

Goertzel 算法

$$
\begin{aligned}
X[k] &= \sum_{l=0}^{N-1} x[l] W_N^{kl} \\
&= W_N^{kN} \sum_{l=0}^{N-1} x[l] W_N^{-k(N-l)} \\
&\xlongequal{W_N^{kN}=1} \sum_{l=0}^{N-1} x[l] W_N^{-k(N-l)} \longleftarrow \text{具有卷积形式}
\end{aligned}
$$

W_N^k 的周期性。

Goertzel 算法

用系统的概念计算 DFT

$$y_k[n] = \sum_{l=0}^{n-1} x[l] W_N^{-k(n-l)}$$

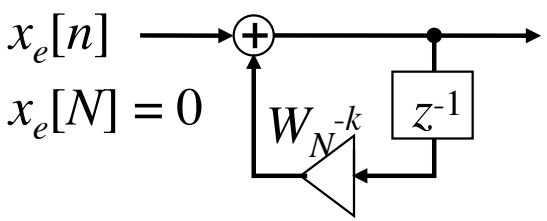
$x_k[n]$ 为有限长因果序列

$$x_e[n] = \begin{cases} x[n], & 0 \le n < N \\ 0, & n = N \end{cases}$$

$h_k[n]$ 为因果序列

$$h_k[n] = \begin{cases} W_N^{-kn}, & n \ge 0 \\ 0, & n < 0 \end{cases}$$

$$y_k[n] = x_e[n] \circledast h_k[n] \Leftrightarrow Y_k(z) = \frac{X_e(z)}{1 - W_N^{-k} z^{-1}}$$



$4N$ 次实数加法
$+ 4N$ 次实数乘法

Goertzel 算法

戈泽尔算法

- 复杂度与直接矩阵运算复杂度一样 $O(N^2)$
  （复杂度略高一些（$4N^2$ 次乘法 $+4N^2$ 次加法），多 $2N$ 次实数加法）
- 不需要事先存储变换矩阵 D_N，降低了存储空间

改进的戈泽尔算法

$$H(z)=\frac{1}{1-W_N^{-k}z^{-1}}=\frac{1-W_N^k z^{-1}}{1-2\cos\frac{2\pi k}{N}z^{-1}+z^{-2}}$$

$$v_k[n]=x_e[n]+2\cos\frac{2\pi k}{N}v_k[n-1]-v_k[n-2]$$

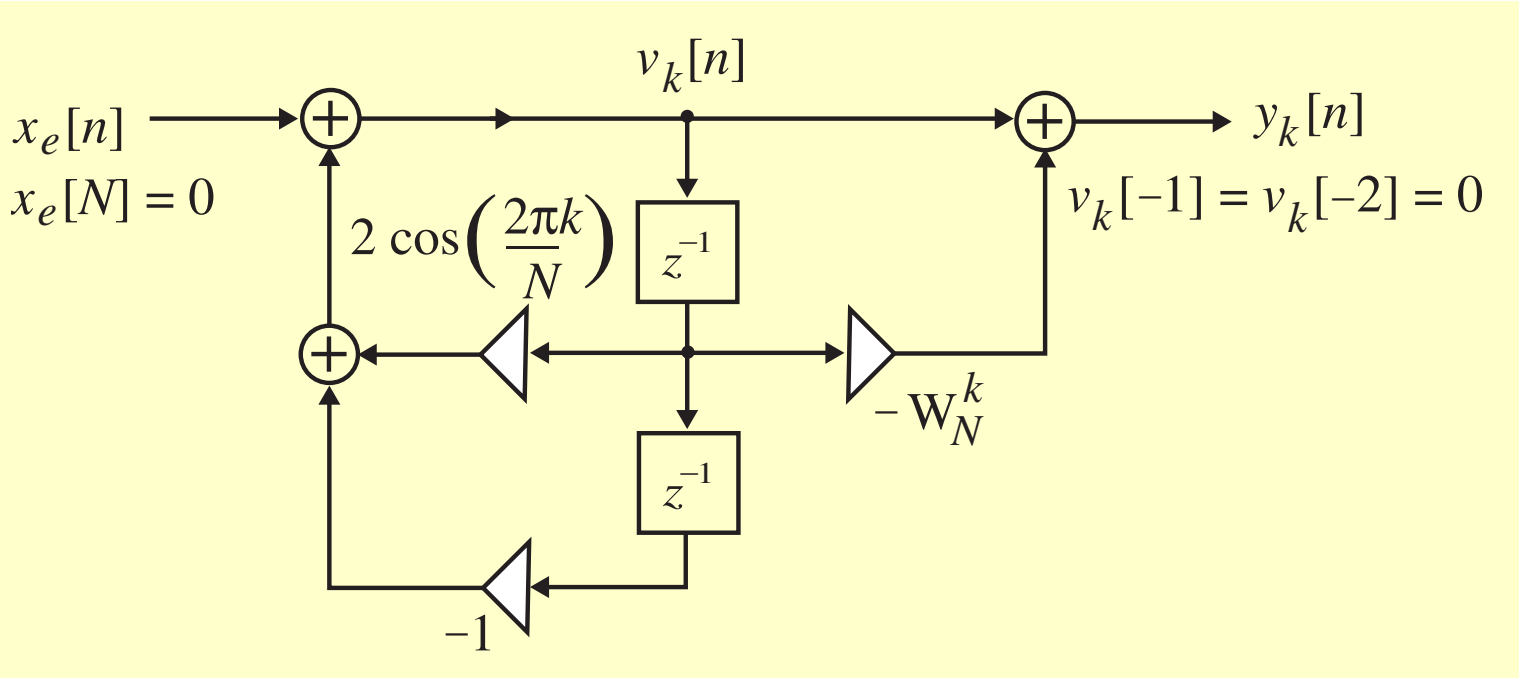
（2 次实数乘法 +4 次实数加法）$\times N$

$$X[k]=y_k[N]=v_k[N]-W_N^k v_k[N-1]$$

4 次实数乘法 +4 次实数加法

- 复杂度为 $(2N^2+4N)$ 次实数乘法 + $(4N^2+4N)$ 次实数加法
  乘法次数大约为直接矩阵运算的一半，加法次数相当
- 进一步化简戈泽尔算法 $\rightarrow$ (N^2+4N) 次实乘 + $(2N^2+4N)$ 次实加

改进的戈泽尔算法


$v_k[n]$
$x_e[n]$
$x_e[N] = 0$
$y_k[n]$
$v_k[-1] = v_k[-2] = 0$
$2\cos\left(\frac{2\pi k}{N}\right)$
z^{-1}
z^{-1}
$-W_N^k$
-1

Outline

基本思想： 化整为零、化零为整

分散、集中和变换，是游击战争灵活使用兵力的三个方法。

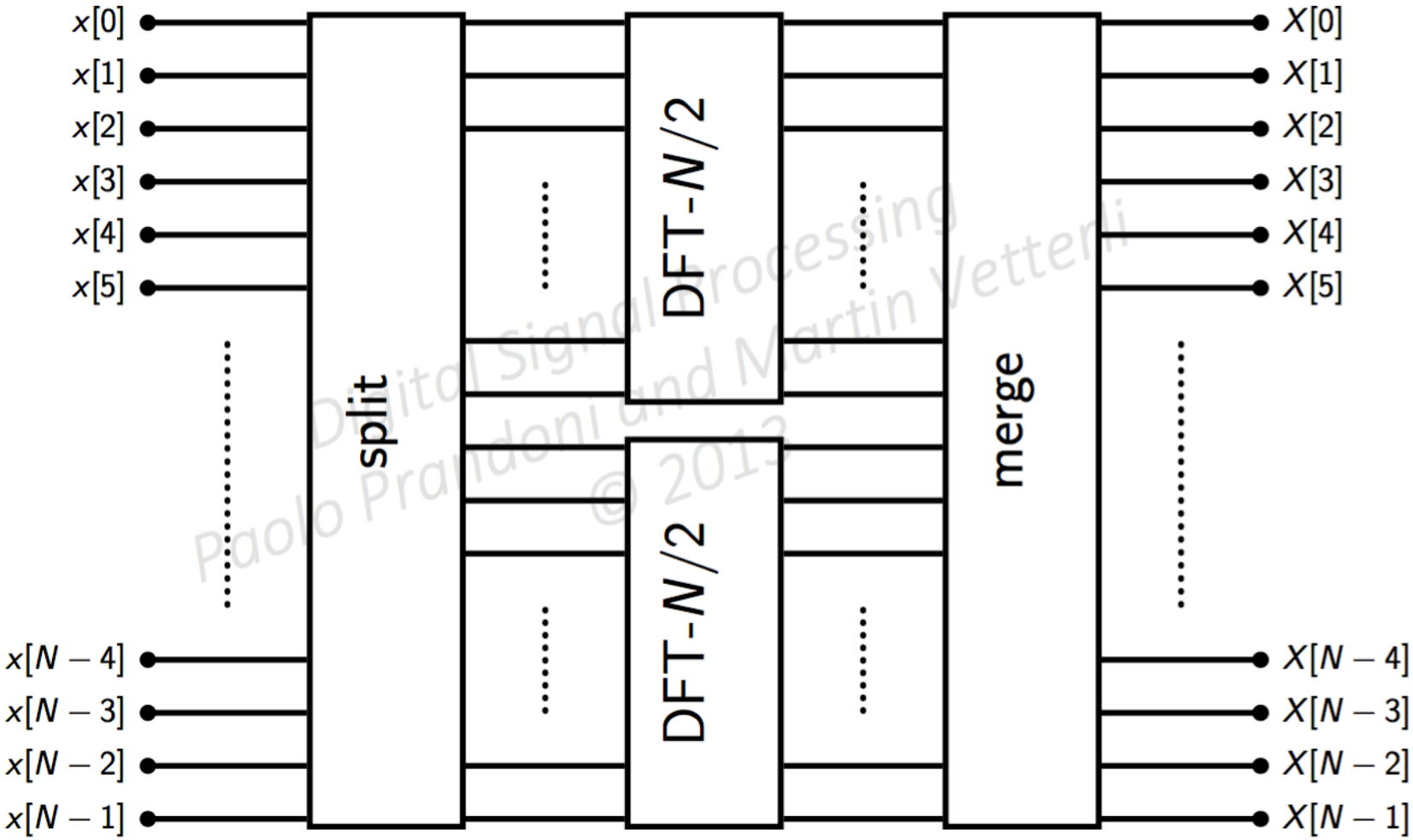
一般地说来，游击队当分散使用，即所谓“化整为零”时，大体上是依下述几种情况实施的……。但不论何种情况，当分散行动时都须注意：（一）保持较大一部分兵力于适当的机动地区，不要绝对地平均分散，一则便于应付可能的事变，一则使分散执行的任务有一个重心；（二）给各分散部队以明确的任务、行动的地区、行动的时期、集合的地点、联络的方法等。

集中使用兵力，即所谓“化零为整”的办法，多半是在敌人进攻之时为了消灭敌人而采取的……。

——《抗日游击战争的战略问题》，毛泽东 *(1938.5)*。

FFT

基本思想： 化整为零、化零为整



时间抽取 (DIT) FFT

- 分解:

$$X[k] = DFT_N\{x[n]\} = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \quad k = 0, 1, ..., N-1$$

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

$$= \sum_{m=0}^{N/2-1} \underbrace{x[2m]}_{x_0[n]} W_{N/2}^{mk} + W_N^k \sum_{m=0}^{N/2-1} \underbrace{x[2m+1]}_{x_1[n]} W_{N/2}^{mk}$$

- DFT 变换的周期性:

$$W_{N/2}^{n(k+N/2)} = W_{N/2}^{nk} \Longrightarrow \begin{cases} X_0[k] & = X_0[k+N/2] \\ X_1[k] & = X_1[k+N/2] \end{cases}$$

时间抽取 (DIT) FFT

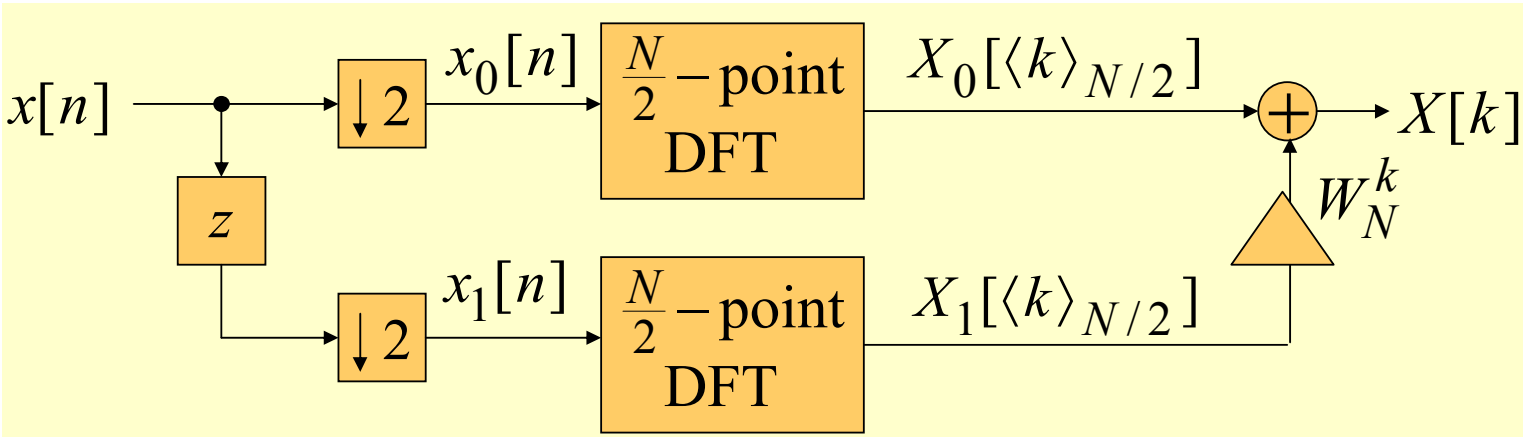
- 分解:

$$
\begin{aligned}
X[k] &= DFT_N\{x[n]\} = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \quad k = 0, 1, ..., N-1 \\
&= \sum_{m=0}^{N/2-1} \left(x[2m] W_N^{2mk} + x[2m+1] W_N^{(2m+1)k} \right) \\
&= \sum_{m=0}^{N/2-1} \underbrace{x[2m]}_{x_0[n]} W_{N/2}^{mk} + W_N^k \sum_{m=0}^{N/2-1} \underbrace{x[2m+1]}_{x_1[n]} W_{N/2}^{mk} \\
&= X_0[\langle k \rangle_{N/2}] + W_N^k X_1[\langle k \rangle_{N/2}], \quad k = 0, 1, ..., N-1
\end{aligned}
$$

- DFT 变换的周期性:

$$
W_{N/2}^{n(k+N/2)} = W_{N/2}^{nk} \Longrightarrow \begin{cases} X_0[k] & = X_0[k+N/2] \\ X_1[k] & = X_1[k+N/2] \end{cases}
$$

时间抽取（DIT）FFT



时间抽取 (DIT) FFT

- **化整为零**：N 点 DFT 变换可以分解成两个 $\frac{N}{2}$ 点的 DFT 变换

$$X[k] = DFT_N\{x[n]\} = G[\langle k \rangle_{N/2}] + W_N^k H[\langle k \rangle_{N/2}], \quad k = 0, 1, ..., N-1$$

  - 分解前后复杂度分析：$DFT_N\{x[n]\} \sim O(N^2)$

    $\Rightarrow DFT_{\frac{N}{2}}\{x_0[n]\} \sim O((N/2)^2) = \frac{1}{4} O(N^2)$

- **化整为零后计算复杂度**：
  - 两个 $N/2$ 点 $DFT \approx N^2/2$ 次复数相乘和 $N^2/2$ 次复数相加
  - 两个 $N/2$ 点 DFT 叠加（$k = 0, ..., N-1$）$\approx N$ 次复乘和 N 次复加
  - 总计：$N^2/2 + N$ 次复数相乘和 $N^2/2 + N$ 次复数相加

$$DFT_{\frac{N}{2}}\{x_0[n]\} + W_N^k DFT_{\frac{N}{2}}\{x_1[n]\} \sim 2 \cdot \frac{1}{4} O(N^2) = \frac{1}{2} O(N^2)$$

时间抽取 (DIT) FFT

- **化整为零**：N 点 DFT 变换可以分解成两个 $\frac{N}{2}$ 点的 DFT 变换

$$X[k] = DFT_N\{x[n]\} = G[\langle k \rangle_{N/2}] + W_N^k H[\langle k \rangle_{N/2}], \quad k = 0, 1, \ldots, N-1$$

  - 分解前后复杂度分析：$DFT_N\{x[n]\} \sim O(N^2)$

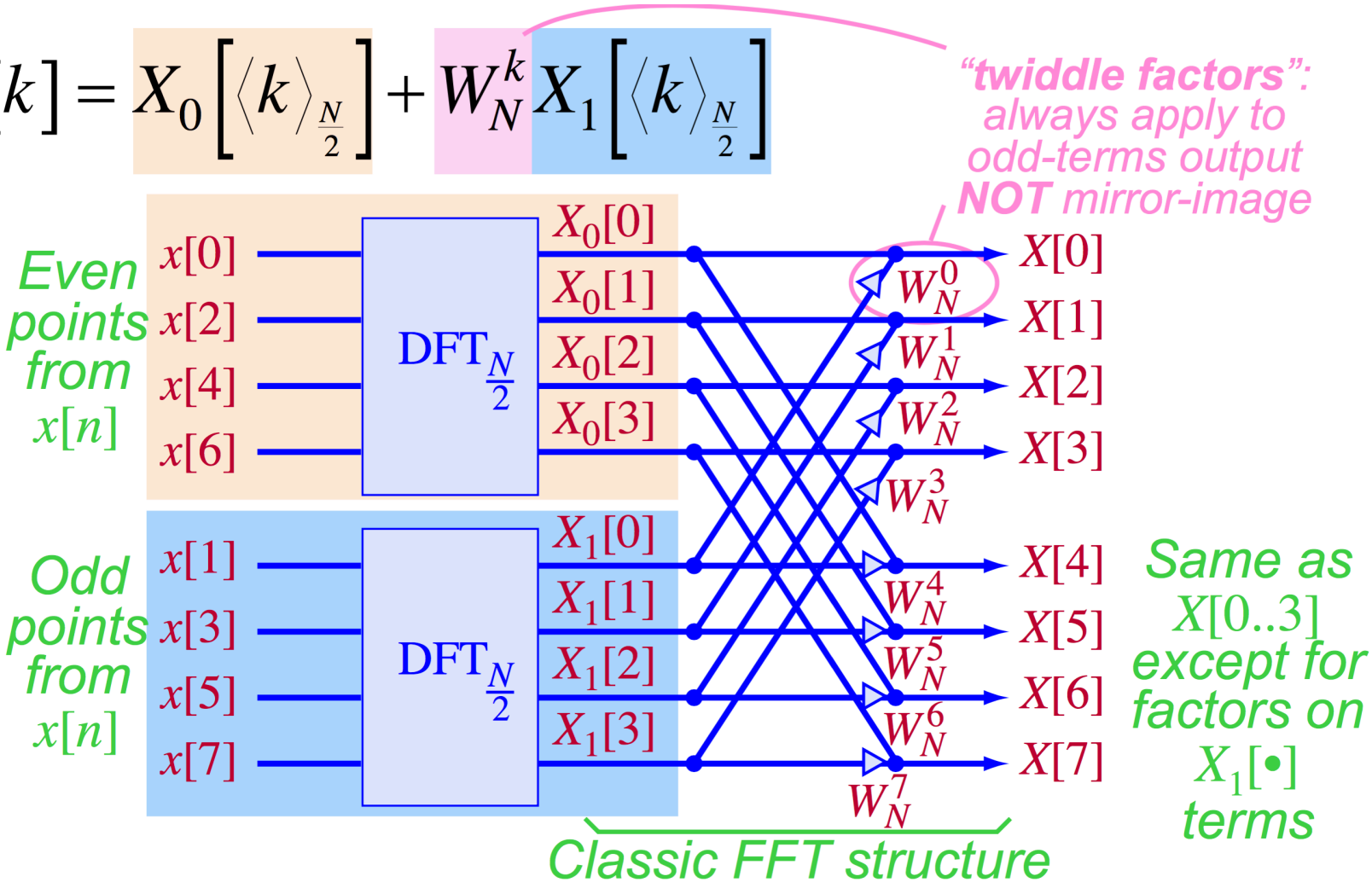
    $\Rightarrow DFT_{\frac{N}{2}}\{x_0[n]\} \sim O((N/2)^2) = \frac{1}{4} O(N^2)$

- **化整为零后计算复杂度**：
  - 两个 $N/2$ 点 $DFT \approx N^2/2$ 次复数相乘和 $N^2/2$ 次复数相加
  - 两个 $N/2$ 点 DFT 叠加（$k = 0, \ldots, N-1$）$\approx N$ 次复乘和 N 次复加
  - 总计：$N^2/2 + N$ 次复数相乘和 $N^2/2 + N$ 次复数相加

$$DFT_{\frac{N}{2}}\{x_0[n]\} + W_N^k DFT_{\frac{N}{2}}\{x_1[n]\} \sim 2 \cdot \frac{1}{4} O(N^2) = \frac{1}{2} O(N^2)$$

One-Stage DIT Flowgraph

$$X[k] = X_0\left[\langle k \rangle_{\frac{N}{2}}\right] + W_N^k X_1\left[\langle k \rangle_{\frac{N}{2}}\right]$$



Two-Stage DIT Stages

- 继续化整为零：

$$DFT_N \to DFT_{N/2} \to DFT_{N/4}$$

  - $N \to N/2$:

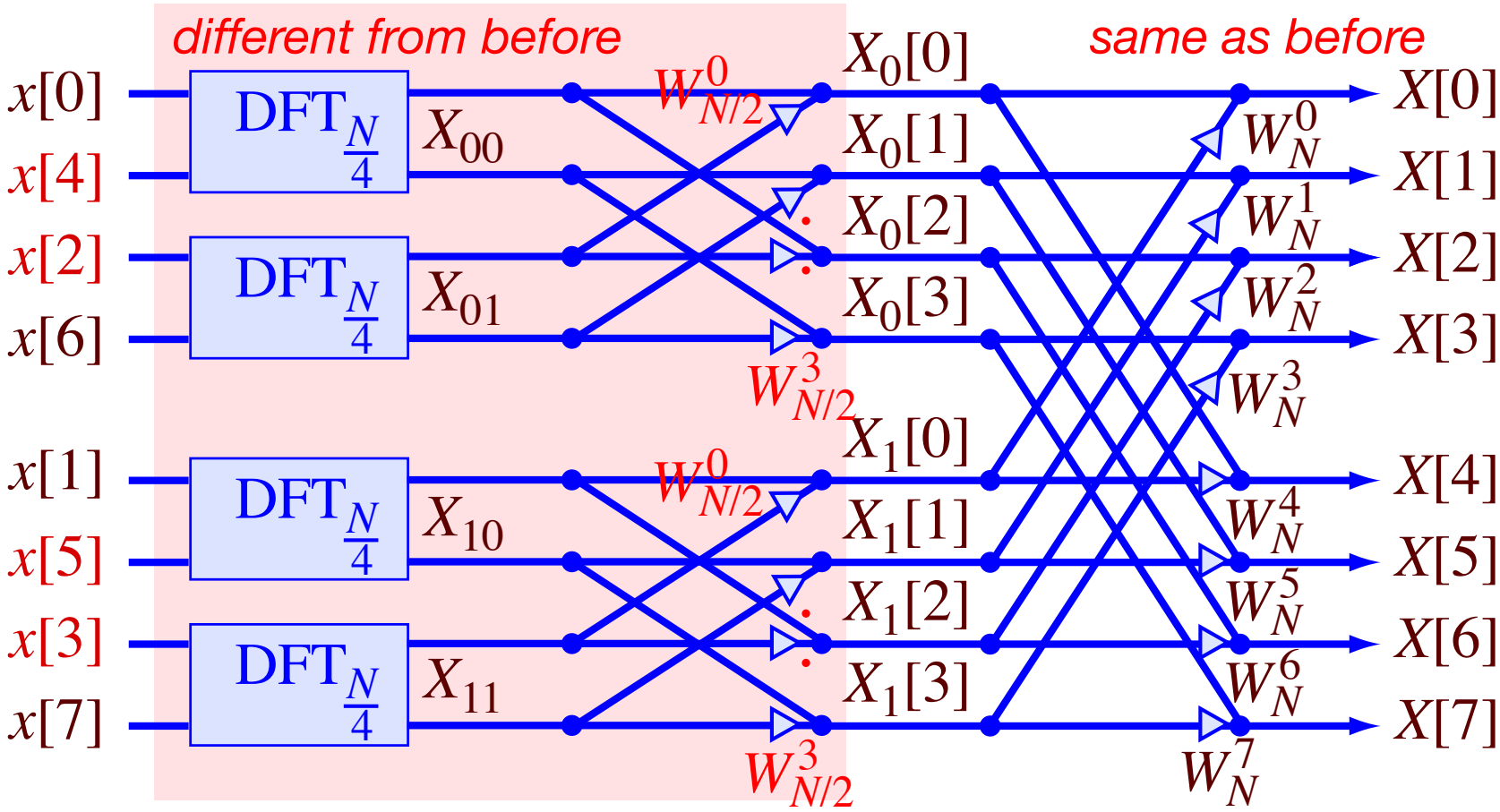
$$X[k] = X_0\left[\langle k\rangle_{N/2}\right] + W_N^k X_1\left[\langle k\rangle_{N/2}\right]$$

  - $N/2 \to N/4$:

$$X_0\left[\langle k\rangle_{N/2}\right] = X_{00}\left[\langle k\rangle_{N/4}\right] + W_{N/2}^k X_{01}\left[\langle k\rangle_{N/4}\right]$$
$$X_1\left[\langle k\rangle_{N/2}\right] = X_{10}\left[\langle k\rangle_{N/4}\right] + W_{N/2}^k X_{11}\left[\langle k\rangle_{N/4}\right]$$

- 复杂度分析：
  - $N^2/4+2N$ 次复数相乘和 $N^2/4+2N$ 次复数相加

Two-Stage DIT Flowgraph



Multi-Stage DIT FFT

- **继续化整为零：**

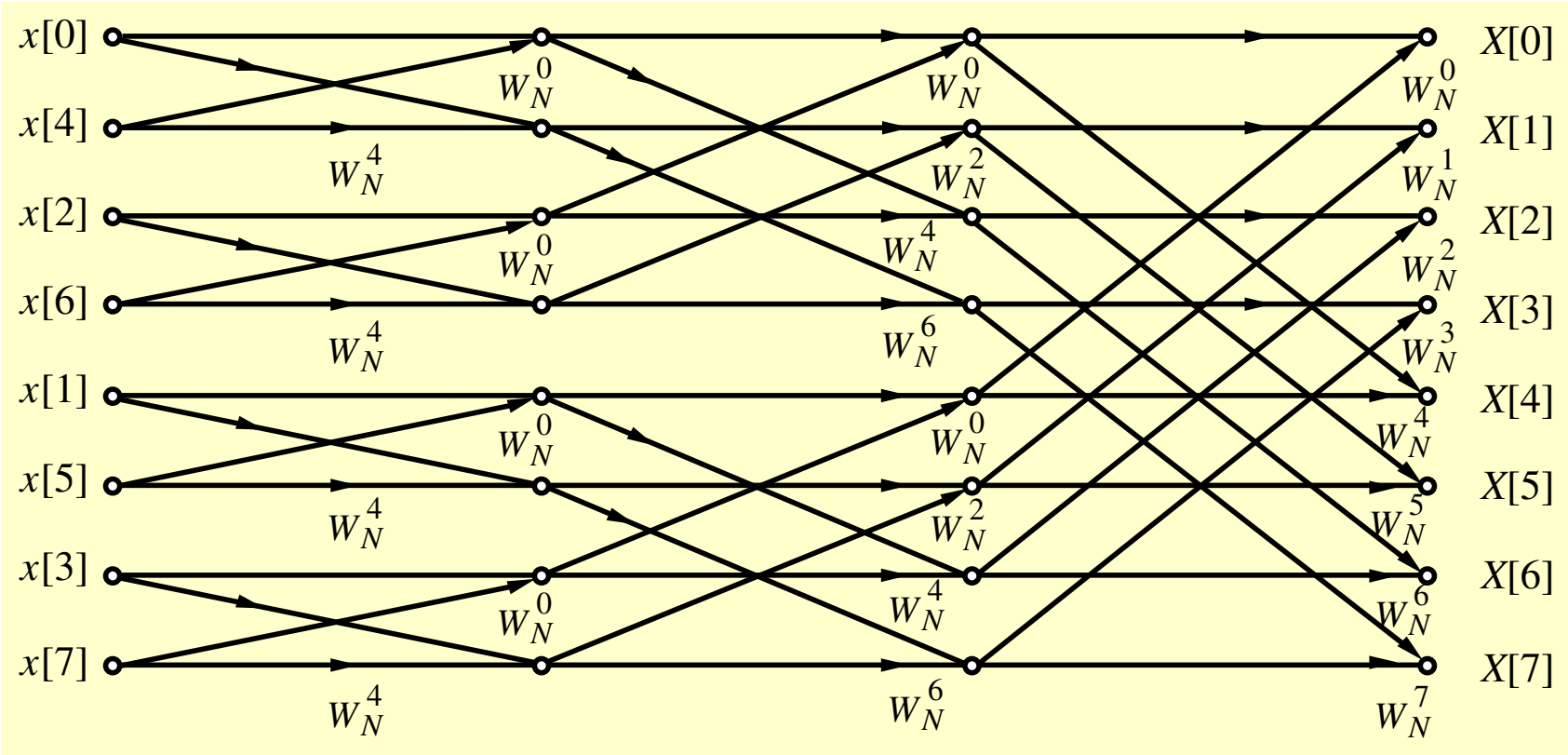
$$DFT_N \to DFT_{N/2} \to DFT_{N/4} \cdots \to DFT_2$$

  - 两点 DFT 变换

$$X[k] = DFT_2\{x[n]\} = \sum_{n=0}^{1} x[n] W_2^k \Rightarrow \begin{cases} X[0] & = x[0] + x[1] \\ X[1] & = x[0] - x[1] \end{cases}$$

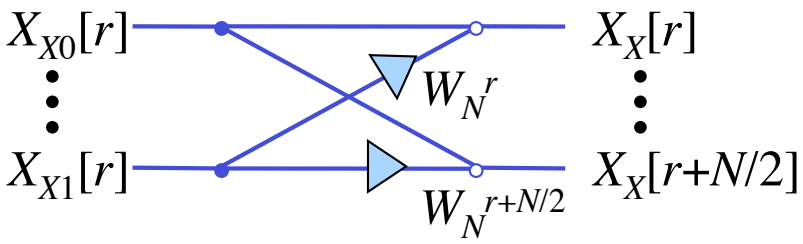
- **复杂度分析：** $(N \triangleq 2^M)$
  - $N^2/2^M + (M-1)N = MN$ 次复乘和 $N^2/2^M + (M-1)N = MN$ 次复加
  - 复杂度：$O(N \cdot \log_2 N)$

Multi-Stage DIT FFT Flowgraph



FFT Implementation Details

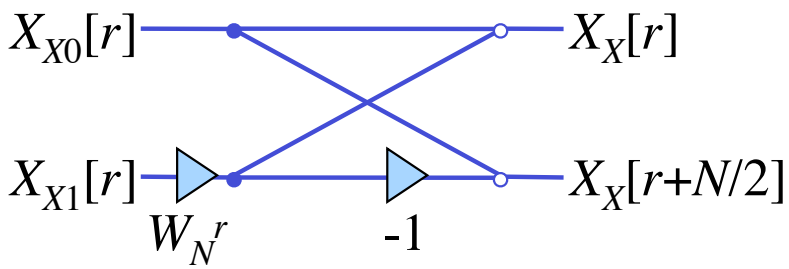
- 蝶形结构：两次复数乘法



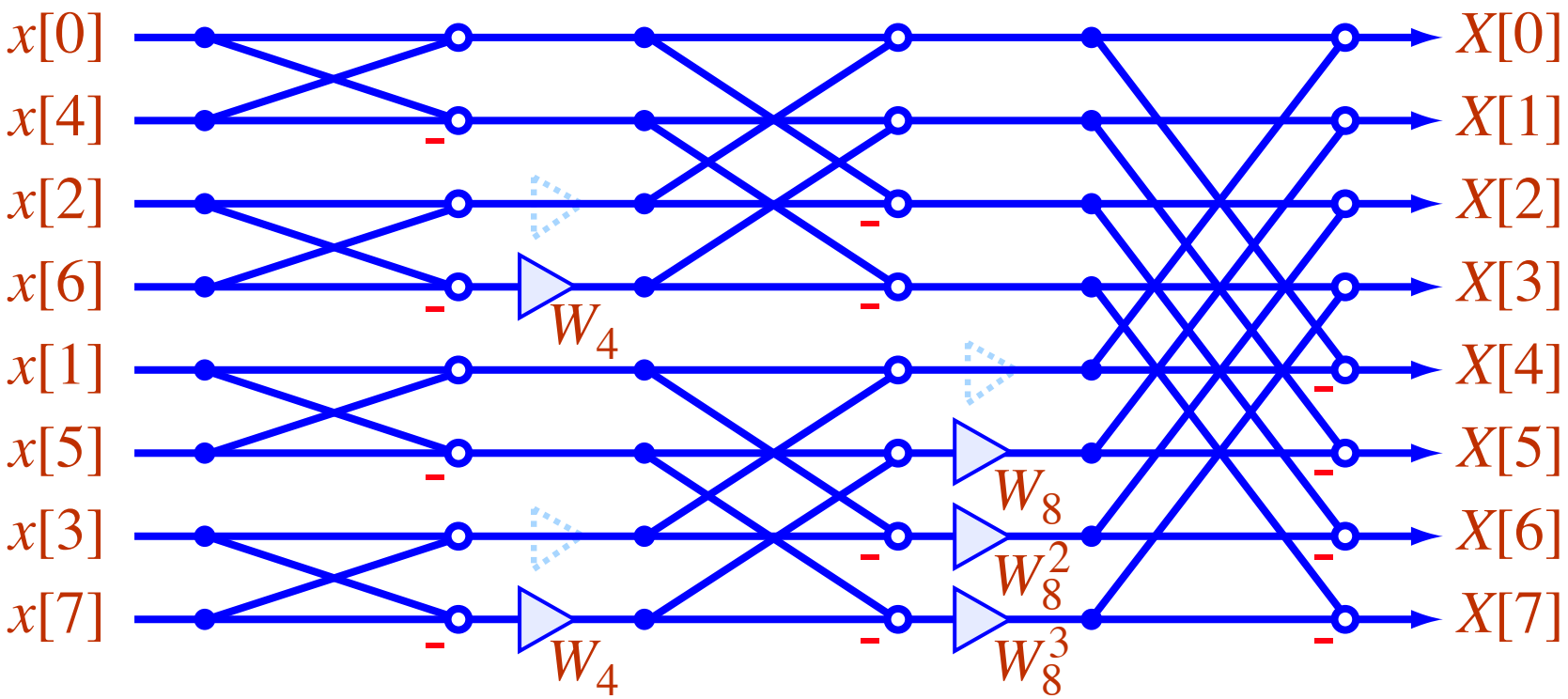
- DFT 变换的对称性：

$$W_N^{r+N/2} = W_N^r W_N^{N/2} = -W_N^r$$

- 简化的蝶形结构：一次复数乘法

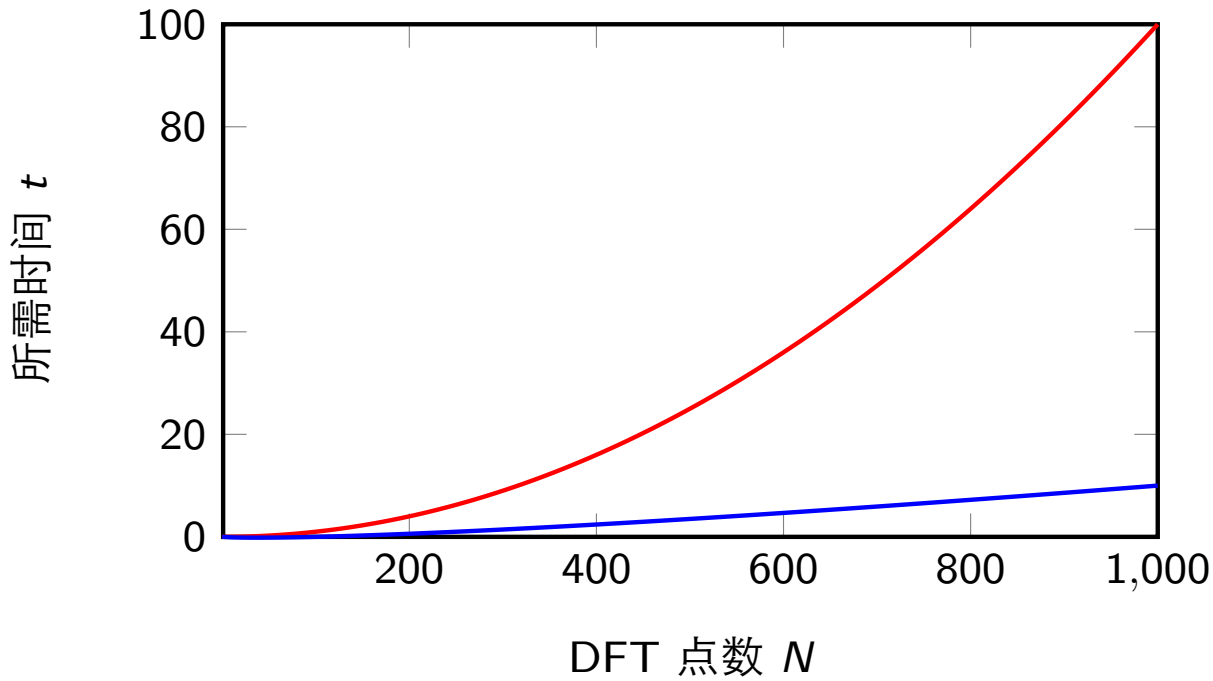


8-pt DIT FFT Flowgraph



Reduce Complexity of DFT

复杂度 $O(N^2) \longrightarrow$ 复杂度 $O(N\log N)$



Radix-R FFT

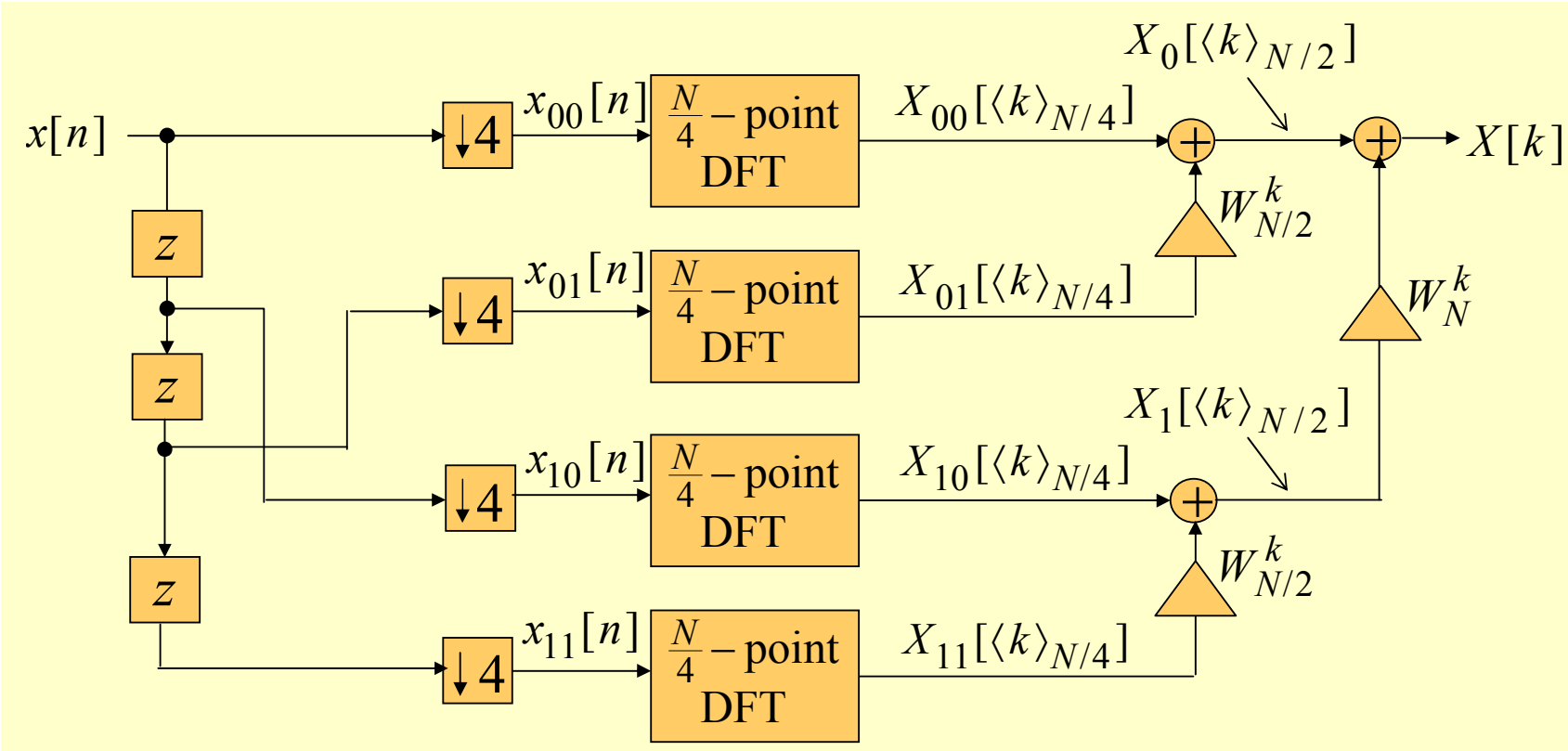
- 如果 $N = 2^M$

$x[n]$:	$x[0]$	$x[1]$	$x[2]$	$x[3]$	$x[4]$	$x[5]$	$x[6]$	$x[7]$
$x_0[n]$:	$x[0]$		$x[2]$		$x[4]$		$x[6]$	
$x_1[n]$:	$x[1]$		$x[3]$		$x[5]$		$x[7]$	

- 如果 $N = 4^M$

$x[n]$:	$x[0]$	$x[1]$	$x[2]$	$x[3]$	$x[4]$	$x[5]$	$x[6]$	$x[7]$
$x_{00}[n]$:	$x[0]$				$x[4]$			
$x_{01}[n]$:	$x[2]$				$x[6]$			
$x_{10}[n]$:	$x[1]$				$x[5]$			
$x_{11}[n]$:	$x[3]$				$x[7]$			

Radix-4 FFT


x[n]
z
z
z
↓4
↓4
↓4
↓4
x00[n]
x01[n]
x10[n]
x11[n]
N/4 – point DFT
N/4 – point DFT
N/4 – point DFT
N/4 – point DFT
X00[⟨k⟩N/4]
X01[⟨k⟩N/4]
X10[⟨k⟩N/4]
X11[⟨k⟩N/4]
X0[⟨k⟩N/2]
X1[⟨k⟩N/2]
W^k_N/2
W^k_N/2
W^k_N
X[k]

FFT for other values of N

如果 N 可以分解成很多个较小的值的乘积

$$N = r_1 \cdot r_2 \cdot r_3 \ldots r_v$$

那么可以采用混合抽样的方式，『化整为零』，书中 452 页图 11.30。
其运算量为：

$$\text{乘法（加法）运算总数} = \left(\sum_{i=1}^{v} r_i - v \right) N$$

Outline

频率抽取（DIF）FFT

- 考虑长度为 $N=2^{\mu}$ 的序列 $x[n]$，其 z 变换为

$$X(z)=\sum_{n=0}^{N-1}x[n]z^{-n}=X_a(z)+z^{-N/2}X_b(z)$$

其中，

$$X_a(z)=\sum_{n=0}^{N/2-1}x[n]z^{-n}$$

$$X_b(z)=\sum_{n=0}^{N/2-1}x[N/2+n]z^{-n}$$

- 根据 z 变换与 DFT 变的关系，在单位圆上等间隔采样，即 $z=W_N^{-k}$

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

频率抽取（DIF）FFT

- 当 k 为偶数

$$X[2\ell] = \sum_{n=0}^{N/2-1} (x[n] + x[N/2+n]) W_N^{2n\ell}, \quad \ell = 0, 1, ..., N/2-1$$
$$= \sum_{n=0}^{N/2-1} (x[n] + x[N/2+n]) W_{N/2}^{n\ell}$$

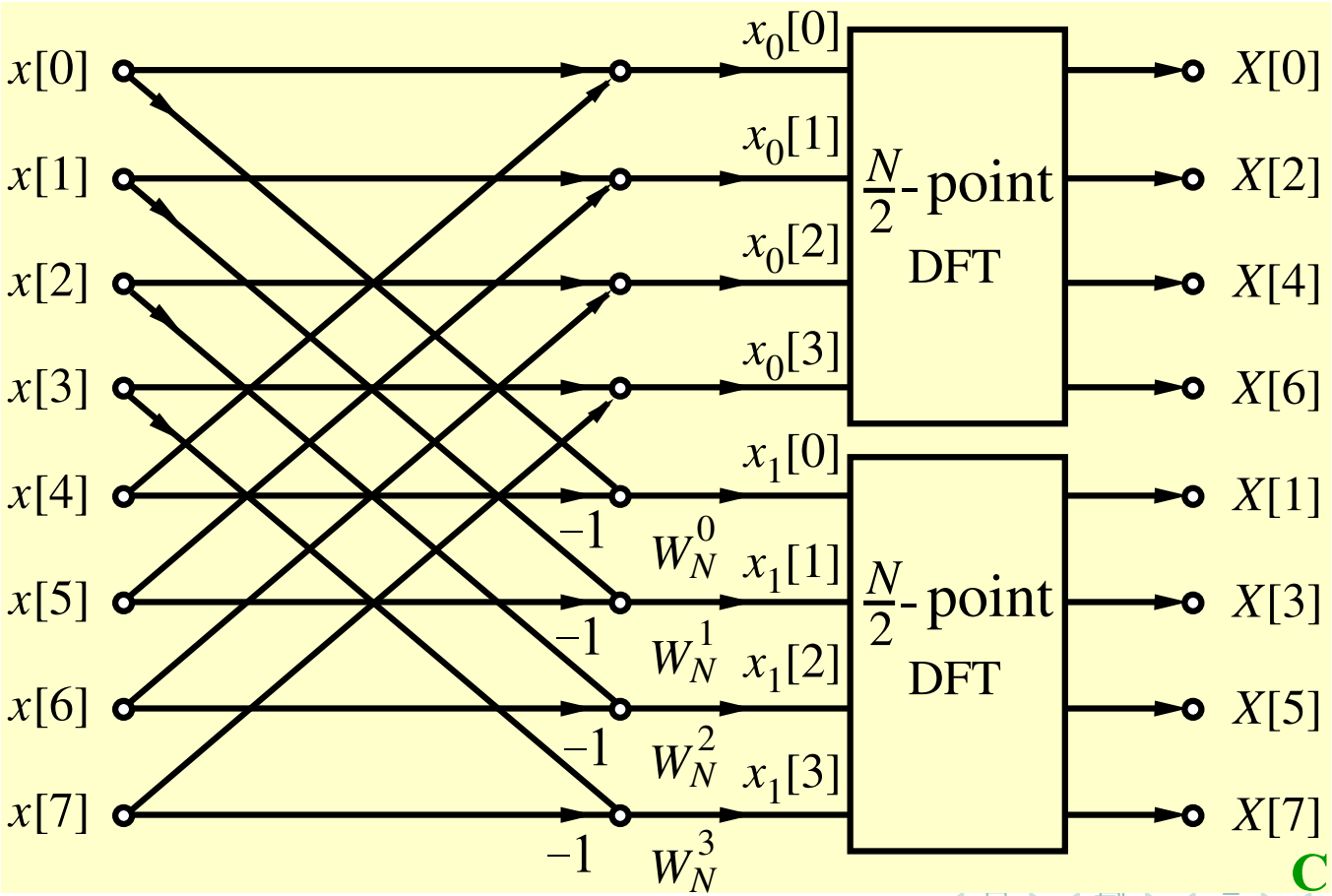
- 当 k 为奇数

$$X[2\ell+1] = \sum_{n=0}^{N/2-1} (x[n] - x[N/2+n]) W_N^{n(2\ell+1)}, \quad \ell = 0, 1, ..., N/2-1$$
$$= \sum_{n=0}^{N/2-1} (x[n] - x[N/2+n]) W_{N/2}^{n\ell} W_N^n$$

频率抽取（DIF）FFT

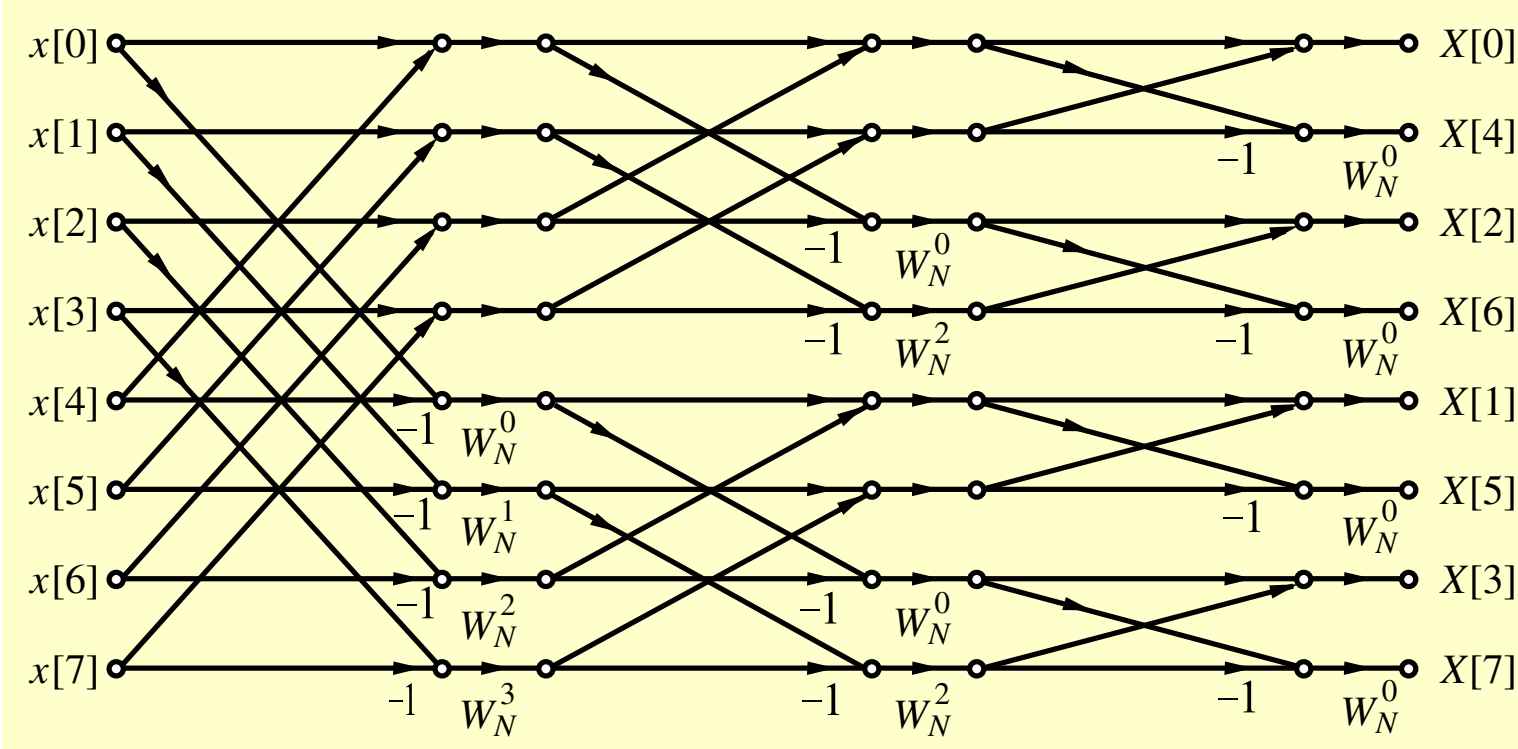
- 频率分解：

$$X[2\ell] = \sum_{n=0}^{N/2-1} x_0[n] W_{N/2}^{n\ell}, \quad X[2\ell+1] = \sum_{n=0}^{N/2-1} x_1[n] W_{N/2}^{n\ell}, \quad \ell = 0, 1, 2, \ldots, N/2-1$$



频率抽取（DIF）FFT

- 继续对 X_0 和 X_1 进行分解：



Outline

Inverse FFT

DFT

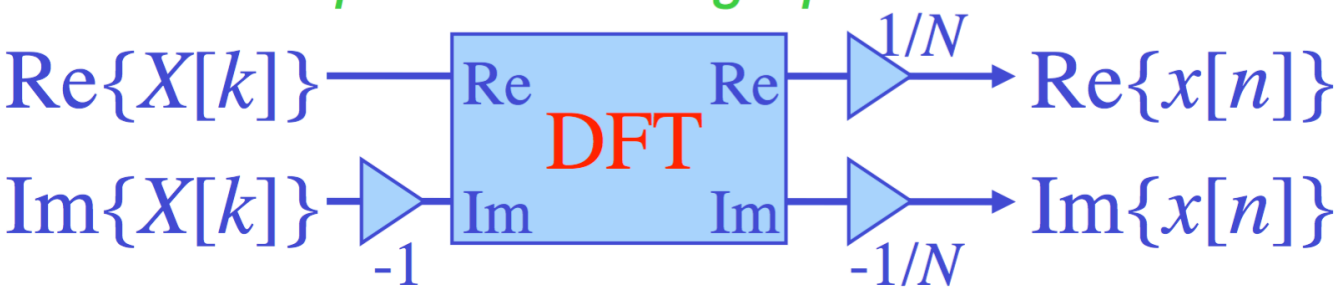
$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}$$

Inverse DFT

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

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

pure real flowgraph



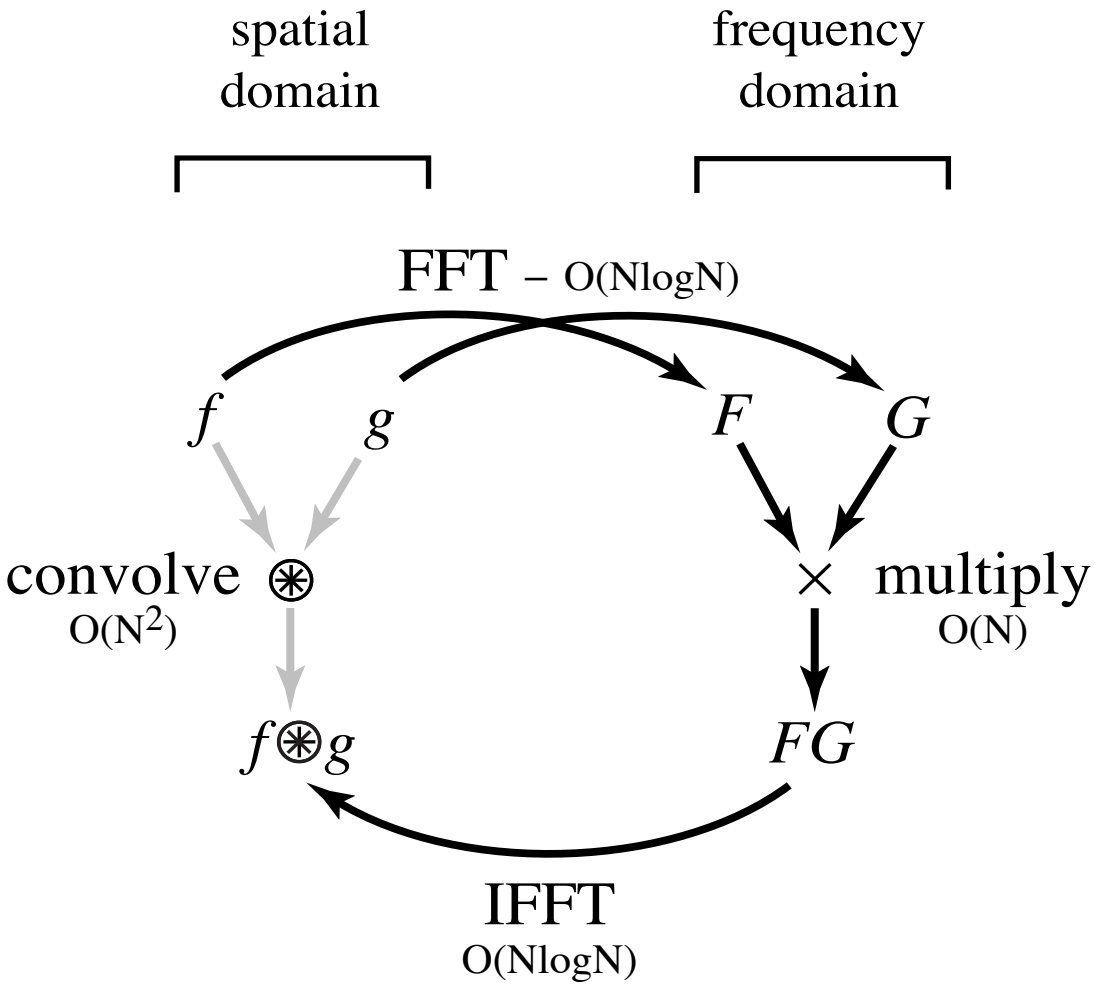
Outline

FFT 的应用

- Efficient matrix-vector multiplication for Toeplitz, circulant and other structured matrices
- Filtering algorithms
- Fast algorithms for discrete cosine or sine transforms (example, Fast DCT used for JPEG, MP3/MPEG encoding)
- ...

[1] Rockmore, D.N. (January 2000). "The FFT: an algorithm the whole family can use". Computing in Science Engineering. 2 (1): 60–64

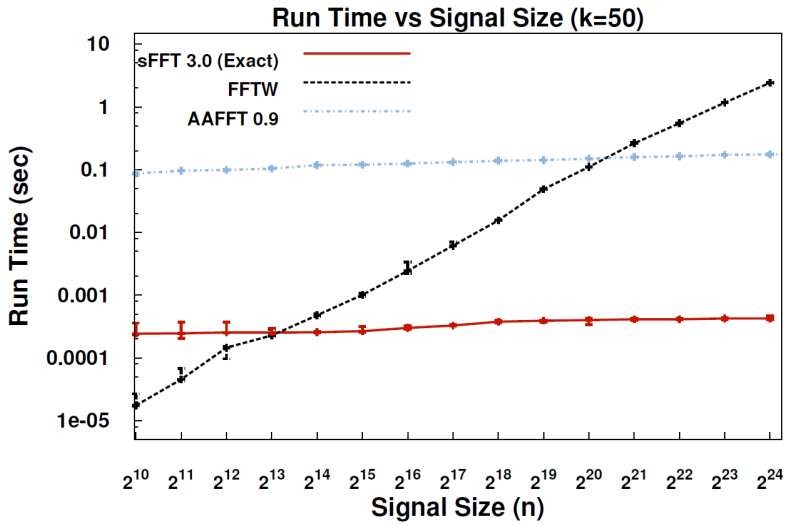
卷积的高效计算


spatial
domain
frequency
domain
FFT – O(NlogN)
f
g
F
G
convolve
O(N²)
multiply
O(N)
f⊛g
FG
IFFT
O(NlogN)

FFT 算法库

- 最快的 FFT 算法库：FFTW，http://www.fftw.org/

- 比 FFT 还要快的 SFFT：Sparse Fast Fourier Transform, http://groups.csail.mit.edu/netmit/sFFT/



FFT 相关的研究领域

- 大数据 FFT：天文观测领域，例如引力波测量（LIGO）中，研究如何对百万甚至亿万点的数据进行 DFT 变换；
- 近似 FFT：MRI 领域，频域采样不一定是等间隔的；
- 量子 FFT：研究 2 进制数据的快速傅里叶变换；
- ...

1. Cormen and Nicol (1998). "Performing out-of-core FFTs on parallel disk systems" (PDF). Parallel Computing. 24 (1): 5–20.
2. Dutt, A.; Rokhlin, V. (November 1, 1993). "Fast Fourier Transforms for Nonequispaced Data". SIAM Journal on Scientific Computing. 14 (6): 1368–1393.
3. L. Hales, S. Hallgren, An improved quantum Fourier transform algorithm and applications, Proceedings of the 41st Annual Symposium on Foundations of Computer Science, p. 515, November 12–14, 2000