

# Digital Signal Processing

## 第3章：快速傅立叶变换

尹华锐

中国科学技术大学 信息科学技术学院

# 教学提纲

- 关于 DFT 的若干总结和讨论
- DFT 走上实用面临的挑战
- 基于时间抽取DIT的快速傅里叶变换
- 基于频率抽取DIF的快速傅里叶变换
- 复合数快速傅里叶变换
- 快速傅里叶变换的应用举例

# DFT 回顾

## CTFT 输入输出特点

- 输入  $x(t)$ , 时域连续; 输出  $X(j\omega)$ , 频域连续

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt$$
$$x(t) = \frac{1}{2j\pi} \int_{-\infty}^{\infty} X(j\omega)e^{j\omega t} d\omega$$

# DFT 回顾

## CTFT 输入输出特点

- 输入  $x(t)$ , 时域连续; 输出  $X(j\omega)$ , 频域连续

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt$$
$$x(t) = \frac{1}{2j\pi} \int_{-\infty}^{\infty} X(j\omega)e^{j\omega t} d\omega$$

## CTFT 计算机处理面临的挑战:

# DFT 回顾

## CTFT 输入输出特点

- 输入  $x(t)$ , 时域连续; 输出  $X(j\omega)$ , 频域连续

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt$$
$$x(t) = \frac{1}{2j\pi} \int_{-\infty}^{\infty} X(j\omega)e^{j\omega t} d\omega$$

## CTFT 计算机处理面临的挑战:

- 时域连续信号如何输入?

# DFT 回顾

## CTFT 输入输出特点

- 输入  $x(t)$ , 时域连续; 输出  $X(j\omega)$ , 频域连续

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt$$
$$x(t) = \frac{1}{2j\pi} \int_{-\infty}^{\infty} X(j\omega)e^{j\omega t} d\omega$$

## CTFT 计算机处理面临的挑战:

- 时域连续信号如何输入?
- 频域连续信号如何输出?

# DFT 回顾

## CTFT 输入输出特点

- 输入  $x(t)$ , 时域连续; 输出  $X(j\omega)$ , 频域连续

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt$$
$$x(t) = \frac{1}{2j\pi} \int_{-\infty}^{\infty} X(j\omega)e^{j\omega t} d\omega$$

## CTFT 计算机处理面临的挑战:

- 时域连续信号如何输入?
- 频域连续信号如何输出?
- 时间域无穷长信号如何处理?

# DFT 回顾

## CTFT 输入输出特点

- 输入  $x(t)$ , 时域连续; 输出  $X(j\omega)$ , 频域连续

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt$$
$$x(t) = \frac{1}{2j\pi} \int_{-\infty}^{\infty} X(j\omega)e^{j\omega t} d\omega$$

## CTFT 计算机处理面临的挑战:

- 时域连续信号如何输入?
- 频域连续信号如何输出?
- 时间域无穷长信号如何处理?

$$X(j\omega) = \int_{t=-\infty}^{\infty} x(t)e^{-j\omega t} dt, \forall \omega \in \mathcal{R}$$



# DFT 回顾

## CTFT 输入输出特点

- 输入  $x(t)$ , 时域连续; 输出  $X(j\omega)$ , 频域连续

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt$$
$$x(t) = \frac{1}{2j\pi} \int_{-\infty}^{\infty} X(j\omega)e^{j\omega t} d\omega$$

## CTFT 计算机处理面临的挑战:

- 时域连续信号如何输入?
- 频域连续信号如何输出?
- 时间域无穷长信号如何处理?

$$X(j\omega) = \int_{t=-\infty}^{\infty} x(t)e^{-j\omega t} dt, \forall \omega \in \mathcal{R}$$

时间利用若干离散点近似计算积分

# DFT 回顾

## CTFT 输入输出特点

- 输入  $x(t)$ , 时域连续; 输出  $X(j\omega)$ , 频域连续

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt$$
$$x(t) = \frac{1}{2j\pi} \int_{-\infty}^{\infty} X(j\omega)e^{j\omega t} d\omega$$

## CTFT 计算机处理面临的挑战:

- 时域连续信号如何输入?
- 频域连续信号如何输出?
- 时间域无穷长信号如何处理?

$$X(j\omega) = \int_{t=-\infty}^{\infty} x(t)e^{-j\omega t} dt, \forall \omega \in \mathcal{R}$$

计算频率域若干离散点的值作为频率域结果

时间利用若干离散点近似计算积分

# DFT 回顾

## CTFT 输入输出特点

- 输入  $x(t)$ , 时域连续; 输出  $X(j\omega)$ , 频域连续

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt$$
$$x(t) = \frac{1}{2j\pi} \int_{-\infty}^{\infty} X(j\omega)e^{j\omega t} d\omega$$

## CTFT 计算机处理面临的挑战:

- 时域连续信号如何输入?
- 频域连续信号如何输出?
- 时间域无穷长信号如何处理?

无穷长时间  
截取一段处理

$$X(j\omega) = \int_{t=-\infty}^{\infty} x(t)e^{-j\omega t} dt, \forall \omega \in \mathcal{R}$$

计算频率域若干离散点的值作为频率域结果

时间利用若干离散点近似计算积分

# 适应性近似改造处理带来的后果

## 处理策略和手段

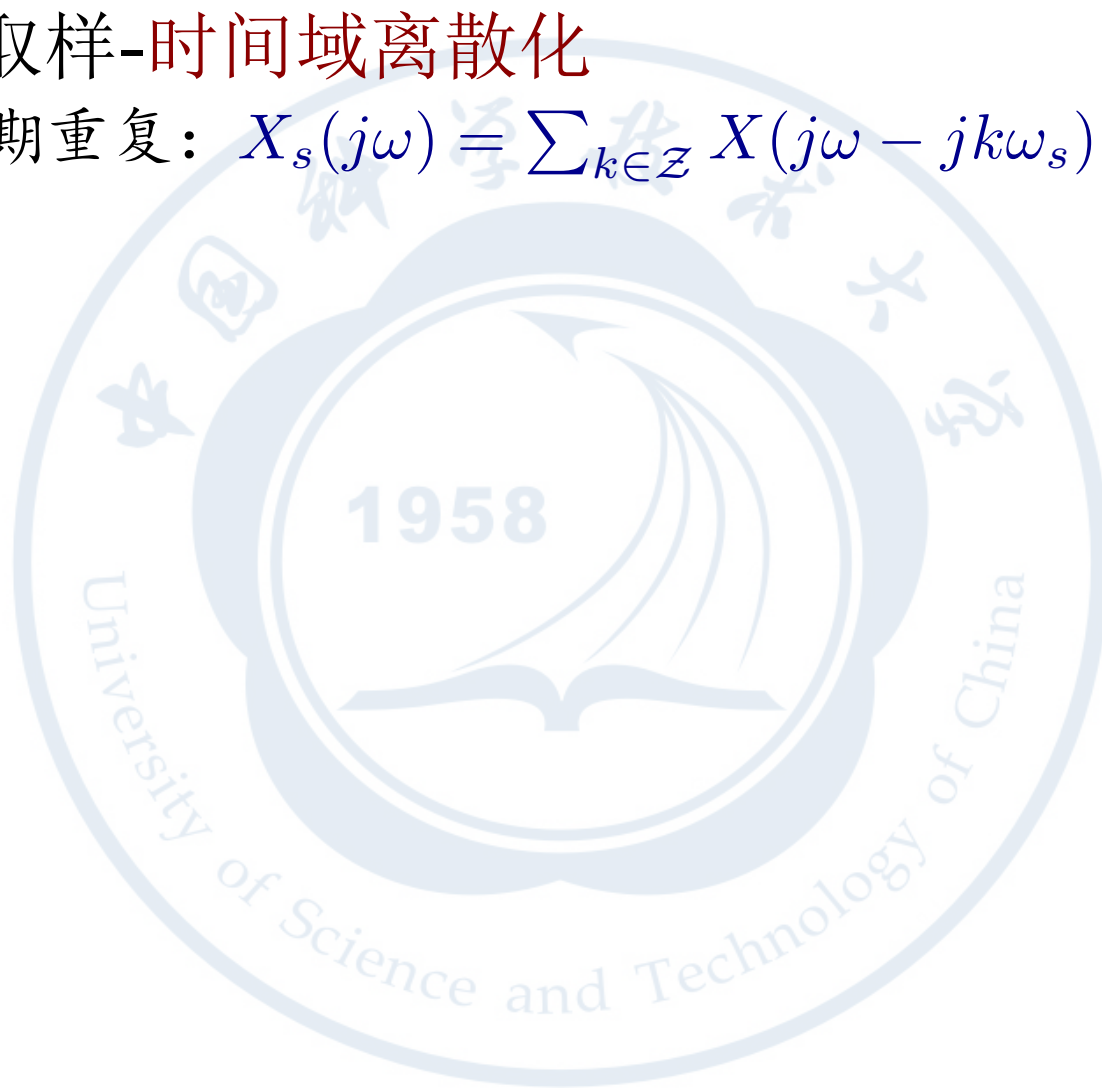


# 适应性近似改造处理带来的后果

## 处理策略和手段

### ■ 时间域取样-时间域离散化

频率域周期重复:  $X_s(j\omega) = \sum_{k \in \mathbb{Z}} X(j\omega - jk\omega_s)$



# 适应性近似改造处理带来的后果

## 处理策略和手段

### ■ 时间域取样-时间域离散化

频率域周期重复:  $X_s(j\omega) = \sum_{k \in \mathbb{Z}} X(j\omega - jk\omega_s)$

### ■ 频率域等间隔取样-频率域离散化

时间域周期重复:  $\tilde{x}(t) = \sum_{m \in \mathbb{Z}} x(t - mT), T = 2\pi/\omega_s$

# 适应性近似改造处理带来的后果

## 处理策略和手段

### ■ 时间域取样-时间域离散化

频率域周期重复:  $X_s(j\omega) = \sum_{k \in \mathbb{Z}} X(j\omega - jk\omega_s)$

### ■ 频率域等间隔取样-频率域离散化

时间域周期重复:  $\tilde{x}(t) = \sum_{m \in \mathbb{Z}} x(t - mT), T = 2\pi/\omega_s$

### ■ 积分（求和）区间有限化-时间域截断

$$x(t) = x(t)(u(t) - u(t - T_{tr})) \rightarrow \tilde{X}(j\omega) = X(j\omega) \otimes \frac{1}{j\omega} (1 - e^{-j\omega T_{tr}})$$

频谱无限展宽



# 适应性近似改造处理带来的后果

## 处理策略和手段

### ■ 时间域取样-时间域离散化

频率域周期重复:  $X_s(j\omega) = \sum_{k \in \mathbb{Z}} X(j\omega - jk\omega_s)$

### ■ 频率域等间隔取样-频率域离散化

时间域周期重复:  $\tilde{x}(t) = \sum_{m \in \mathbb{Z}} x(t - mT), T = 2\pi/\omega_s$

### ■ 积分（求和）区间有限化-时间域截断

$$x(t) = x(t)(u(t) - u(t - T_{tr})) \rightarrow \tilde{X}(j\omega) = X(j\omega) \otimes \frac{1}{j\omega} (1 - e^{-j\omega T_{tr}})$$

频谱无限展宽

频域周期化 + 频谱无限展宽  $\rightarrow$  混叠不可避免



# 适应性近似改造处理带来的后果

## 处理策略和手段

### ■ 时间域取样-时间域离散化

频率域周期重复:  $X_s(j\omega) = \sum_{k \in \mathbb{Z}} X(j\omega - jk\omega_s)$

### ■ 频率域等间隔取样-频率域离散化

时间域周期重复:  $\tilde{x}(t) = \sum_{m \in \mathbb{Z}} x(t - mT), T = 2\pi/\omega_s$

### ■ 积分（求和）区间有限化-时间域截断

$$x(t) = x(t)(u(t) - u(t - T_{tr})) \rightarrow \tilde{X}(j\omega) = X(j\omega) \otimes \frac{1}{j\omega} (1 - e^{-j\omega T_{tr}})$$

频谱无限展宽

频域周期化 + 频谱无限展宽  $\rightarrow$  混叠不可避免

频率域取样  $\rightarrow$  栅栏效应

# 适应性近似改造处理带来的后果

## 处理策略和手段

### ■ 时间域取样-时间域离散化

频率域周期重复:  $X_s(j\omega) = \sum_{k \in \mathbb{Z}} X(j\omega - jk\omega_s)$

### ■ 频率域等间隔取样-频率域离散化

时间域周期重复:  $\tilde{x}(t) = \sum_{m \in \mathbb{Z}} x(t - mT), T = 2\pi/\omega_s$

### ■ 积分（求和）区间有限化-时间域截断

$$x(t) = x(t)(u(t) - u(t - T_{tr})) \rightarrow \tilde{X}(j\omega) = X(j\omega) \otimes \frac{1}{j\omega} (1 - e^{-j\omega T_{tr}})$$

频谱无限展宽

频域周期化 + 频谱无限展宽  $\rightarrow$  混叠不可避免

频率域取样  $\rightarrow$  栅栏效应

栅栏效应 + 时间域周期重复  $\rightarrow$  频谱泄露

# 适应性改造必须面临的参数约束

■ 时间域取样  $\Rightarrow T > \frac{1}{2B_{max}}$



# 适应性改造必须面临的参数约束

- 时间域取样  $\Rightarrow T > \frac{1}{2B_{max}}$
- 时间记录长度  $\Rightarrow T_{rec} > \frac{1}{F}$ ,  $F$  频率采样间隔, 频谱分辨率



# 适应性改造必须面临的参数约束

- 时间域取样  $\Rightarrow T > \frac{1}{2B_{max}}$
- 时间记录长度  $\Rightarrow T_{rec} > \frac{1}{F}$ ,  $F$  频率采样间隔, 频谱分辨率
- 频谱泄露  $\Rightarrow$  窗函数, 抑制不连续点

# 适应性改造必须面临的参数约束

- 时间域取样  $\Rightarrow T > \frac{1}{2B_{max}}$
- 时间记录长度  $\Rightarrow T_{rec} > \frac{1}{F}$ ,  $F$  频率采样间隔, 频谱分辨率
- 频谱泄露  $\Rightarrow$  窗函数, 抑制不连续点

Rectangle, Triangle, Hamming, Hanning and Blackman Window

不同窗函数具有不同的频谱泄漏抑制能力的物理解释是时间域周期重复的链接点的连续性质。

# 适应性改造必须面临的参数约束

- 时间域取样  $\Rightarrow T > \frac{1}{2B_{max}}$
- 时间记录长度  $\Rightarrow T_{rec} > \frac{1}{F}$ ,  $F$  频率采样间隔, 频谱分辨率
- 频谱泄露  $\Rightarrow$  窗函数, 抑制不连续点

Rectangle, Triangle, Hamming, Hanning and Blackman Window

不同窗函数具有不同的频谱泄漏抑制能力的物理解释是时间域周期重复的链接点的连续性质。

## 无止境的提高窗函数性能是否有意义?

实际上频谱泄漏能量小于噪声能量再优化没有任何价值



# DFT实际运用面临的实现问题

## 系统实现两大挑战:

■ 时间复杂度

■ 空间复杂度

## DFT 计算公式:

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

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



# DSP实际运用面临的复杂度问题

## 运算复杂度:

- 乘法:  $N^2$  复数乘法
- 加法:  $N(N-1)$  复数加法

## 空间复杂度:

- 输入:  $N$  复数
- 输出:  $N$  复数
- 旋转因子:  $N/4$

能否进一步减少?

# DSP面临的复杂度问题

## Toy Example:

一个无线信号进行频谱分析, 信号带宽为 1MHz, 频谱分辨率为 10Hz, 实时频谱分析所需要的运算能力.

$$t_p = 1/10\text{Hz} = 0.1, T_s = 1/2B = 0.5\mu s$$

$$N = t_p/T_s = 2 \times 10^5$$

每次处理乘法运算次数:  $4 \times 10^{10} = 40G(\text{Complex Multi})$

加法运算次数:  $40G(\text{Complex Additive})$

实时运算复杂度: 400G乘法, 400G加法

这个计算复杂度需要多强的计算能力来处理?

# DFT面临的处理复杂度

## Toy Example(Cont)

- Intel X9960 CPU: 16 Cores , CPU running clock @3.2GHz, 900 US Dollars per Chip, 功耗 130Watts
- Peak MMAC:  $3.2\text{G} \times 16 = 51.2\text{GMMACs}$
- 需要 8 片 X9960 Intel 最新的 16 核处理器理论上才能完成该工作

并发传输效率问题, Cache Missing 问题, SDRAM refresh 问题, 还需要更多的芯片才能完成这个工作

# DFT面临的处理复杂度

## Toy Example(Cont)

- Intel X9960 CPU: 16 Cores , CPU running clock @3.2GHz, 900 US Dollars per Chip, 功耗 130Watts
- Peak MMAC:  $3.2\text{G} \times 16 = 51.2\text{GMMACs}$
- 需要 8 片 X9960 Intel 最新的 16 核处理器理论上才能完成该工作

并发传输效率问题, Cache Missing 问题, SDRAM refresh 问题, 还需要更多的芯片才能完成这个工作

**DFT走向实际应用还需要解决实现复杂度问题!**

# DFT面临的处理复杂度

## Toy Example(Cont)

- Intel X9960 CPU: 16 Cores , CPU running clock @3.2GHz, 900 US Dollars per Chip, 功耗 130Watts
- Peak MMAC:  $3.2\text{G} \times 16 = 51.2\text{GMMACs}$
- 需要 8 片 X9960 Intel 最新的 16 核处理器理论上才能完成该工作

并发传输效率问题, Cache Missing 问题, SDRAM refresh 问题, 还需要更多的芯片才能完成这个工作

**DFT走向实际应用还需要解决实现复杂度问题!**

时间复杂度, 空间复杂度, 目前更多考虑的是前者

# DFT解决复杂度的思路

$$\begin{bmatrix} W_N^{0 \times 0} & \dots & W_N^{0 \times j_1} & \dots & W_N^{0 \times j_2} & \dots & W_N^{0 \times N-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ W_N^{k_1 \times 0} & \dots & W_N^{k_1 \times n_1} & \dots & W_N^{k_1 \times n_2} & \dots & W_N^{k_1 \times N} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ W_N^{k_2 \times 0} & \dots & W_N^{k_2 \times n_1} & \dots & W_N^{k_2 \times n_2} & \dots & W_N^{k_2 \times N} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ W_N^{(N-1) \times 0} & \dots & W_N^{(N-1) \times n_1} & \dots & W_N^{(N-1) \times n_2} & \dots & W_N^{(N-1) \times N} \end{bmatrix}$$



# DFT解决复杂度的思路

$$\begin{bmatrix} W_N^{0 \times 0} & \dots & W_N^{0 \times j_1} & \dots & W_N^{0 \times j_2} & \dots & W_N^{0 \times N-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ W_N^{k_1 \times 0} & \dots & W_N^{k_1 \times n_1} & \dots & W_N^{k_1 \times n_2} & \dots & W_N^{k_1 \times N} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ W_N^{k_2 \times 0} & \dots & W_N^{k_2 \times n_1} & \dots & W_N^{k_2 \times n_2} & \dots & W_N^{k_2 \times N} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ W_N^{(N-1) \times 0} & \dots & W_N^{(N-1) \times n_1} & \dots & W_N^{(N-1) \times n_2} & \dots & W_N^{(N-1) \times N} \end{bmatrix}$$

$$k_1 + k_2 = N$$

$$X(k_1) : W_N^{k_1 \times n_1} x(n_1)$$

$$X(k_2) : W_N^{k_2 \times n_1} x(n_1)$$

旋转因子共轭复数，和同一个数相乘，实数乘法完全一样

# DFT解决复杂度的思路

$$\begin{bmatrix} W_N^{0 \times 0} & \dots & W_N^{0 \times j_1} & \dots & W_N^{0 \times j_2} & \dots & W_N^{0 \times N-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ W_N^{k_1 \times 0} & \dots & W_N^{k_1 \times n_1} & \dots & W_N^{k_1 \times n_2} & \dots & W_N^{k_1 \times N} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ W_N^{k_2 \times 0} & \dots & W_N^{k_2 \times n_1} & \dots & W_N^{k_2 \times n_2} & \dots & W_N^{k_2 \times N} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ W_N^{(N-1) \times 0} & \dots & W_N^{(N-1) \times n_1} & \dots & W_N^{(N-1) \times n_2} & \dots & W_N^{(N-1) \times N} \end{bmatrix}$$

$$n_1 + n_2 = N \Rightarrow W_N^{k_1 \times n_1} = \text{conj}(W_N^{k_1 \times n_2})$$

$W_N^{i_1 \times j_1}$  与  $W_N^{i_1 \times j_2}$  共轭复数, 与不同的数相乘后相加, 实部虚部分别处理, 可利用分配律降低乘法次数



# DFT解决复杂度的思路

$$\begin{bmatrix}
 W_N^{0 \times 0} & \dots & W_N^{0 \times j_1} & \dots & W_N^{0 \times j_2} & \dots & W_N^{0 \times N-1} \\
 \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
 W_N^{k_1 \times 0} & \dots & \boxed{W_N^{k_1 \times n_1}} & \dots & \boxed{W_N^{k_1 \times n_2}} & \dots & W_N^{k_1 \times N} \\
 \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
 W_N^{k_2 \times 0} & \dots & \boxed{W_N^{k_2 \times n_1}} & \dots & \boxed{W_N^{k_2 \times n_2}} & \dots & W_N^{k_2 \times N} \\
 \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
 W_N^{(N-1) \times 0} & \dots & W_N^{(N-1) \times n_1} & \dots & W_N^{(N-1) \times n_2} & \dots & W_N^{(N-1) \times N}
 \end{bmatrix}$$

$$((k_1 - k_2) \times n_1) \% N = 0 \rightarrow W_N^{k_1 \times n_1} x(n_1) = W_N^{k_2 \times n_1} x(n_1)$$

$$((k_1 - k_2) \times n_2) \% N = 0 \rightarrow W_N^{k_1 \times n_2} x(n_2) = W_N^{k_2 \times n_2} x(n_2)$$

# DFT解决复杂度的思路

$$\begin{bmatrix}
 W_N^{0 \times 0} & \dots & W_N^{0 \times j_1} & \dots & W_N^{0 \times j_2} & \dots & W_N^{0 \times N-1} \\
 \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
 W_N^{k_1 \times 0} & \dots & \boxed{W_N^{k_1 \times n_1}} & \dots & \boxed{W_N^{k_1 \times n_2}} & \dots & W_N^{k_1 \times N} \\
 \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
 W_N^{k_2 \times 0} & \dots & \boxed{W_N^{k_2 \times n_1}} & \dots & \boxed{W_N^{k_2 \times n_2}} & \dots & W_N^{k_2 \times N} \\
 \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
 W_N^{(N-1) \times 0} & \dots & W_N^{(N-1) \times n_1} & \dots & W_N^{(N-1) \times n_2} & \dots & W_N^{(N-1) \times N}
 \end{bmatrix}$$

$$(k_1 \times (n_1 - n_2)) \% N = 0$$

$$\rightarrow W_N^{k_1 \times n_1} x(n_1) + W_N^{k_1 \times n_2} x(n_2) = W_N^{k_1 \times n_1} (x(n_1) + x(n_2))$$

$$(k_2 \times (n_1 - n_2)) \% N = 0$$

$$\rightarrow W_N^{k_2 \times n_1} x(n_1) + W_N^{k_2 \times n_2} x(n_2) = W_N^{k_2 \times n_1} (x(n_1) + x(n_2))$$

# DFT解决复杂度的思路

$$\begin{bmatrix} W_N^{0 \times 0} & \dots & W_N^{0 \times j_1} & \dots & W_N^{0 \times j_2} & \dots & W_N^{0 \times N-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ W_N^{k_1 \times 0} & \dots & W_N^{k_1 \times n_1} & \dots & W_N^{k_1 \times n_2} & \dots & W_N^{k_1 \times N} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ W_N^{k_2 \times 0} & \dots & W_N^{k_2 \times n_1} & \dots & W_N^{k_2 \times n_2} & \dots & W_N^{k_2 \times N} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ W_N^{(N-1) \times 0} & \dots & W_N^{(N-1) \times n_1} & \dots & W_N^{(N-1) \times n_2} & \dots & W_N^{(N-1) \times N} \end{bmatrix}$$

如何利用旋转因子的周期性，对称性降低重复运算次数，进一步突破制约 **DFT** 走向实用的计算复杂度问题

# 基于时间抽取 (DIT) 的快速傅里叶变换

考虑这样的模型:  $(k_1 - k_2) = N/2, n$  是偶数的情况

$W_N^{k_1 \times n} x(n) = W_N^{k_2 \times n} x(n)$ , 所以  $k_1 - k_2 = N/2$  时,  
 $X(k_1), X(k_2)$  计算过程有大量的重复

我们关注  $X(k), X(k + N/2), 0 \leq k \leq N/2 - 1$ :

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

注意  $n = 2r$  的部分  $X(k), X(k + N/2)$  重复计算! 分离  $n = 2r, n = 2r + 1, 0 \leq r \leq N/2 - 1$

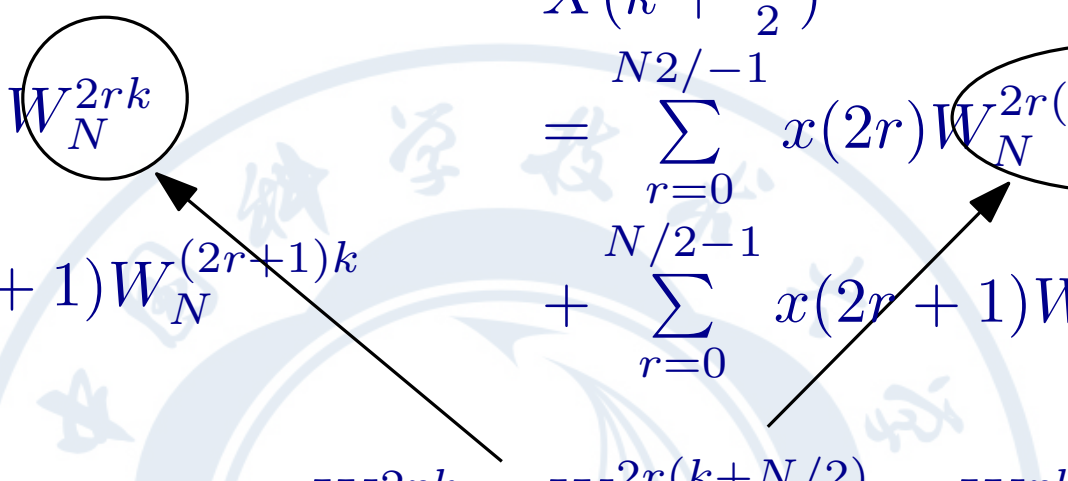
# 基于时间抽取 (DIT) 的快速傅里叶变换

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

# 基于时间抽取 (DIT) 的快速傅里叶变换

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

$W_N^{2rk} = W_N^{2r(k+N/2)} = W_{N/2}^{rk}$



# 基于时间抽取 (DIT) 的快速傅里叶变换

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

$$W_N^{(2r+1)k} = W_N^k W_{\frac{N}{2}}^{rk}$$

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

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



# 基于时间抽取 (DIT) 的快速傅里叶变换

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

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

$$W_N^{2rk} = W_N^{2r(k+N/2)} = W_{N/2}^{rk}$$

$$W_N^{(2r+1)k} = W_N^k W_{N/2}^{rk}$$

$$W_N^{(2r+1)(k+N/2)} = -W_N^k W_{N/2}^{rk}$$

$W^{k+N/2} = -W^k$ ! 这是能够将不符合简化运算的列也转化为简化运算的关键!  $x(2r+1)W_N^{(2r+1)k}$  和  $x(2r+1)W_N^{(2r+1)(k+N/2)}$  无需重复计算!



# 基于时间抽取 (DIT) 的快速傅里叶变换

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

$W_N^{2rk} = W_N^{2r(k+N/2)} = W_{N/2}^{rk}$

$W_N^{(2r+1)k} = W_N^k W_{N/2}^{rk}$ 
 $W_N^{(2r+1)(k+N/2)} = -W_N^k W_{N/2}^{rk}$

$W^{k+N/2} = -W^k$ ! 这是能够将不符合简化运算的列也转化为简化运算的关键!  $x(2r+1)W_N^{(2r+1)k}$  和  $x(2r+1)W_N^{(2r+1)(k+N/2)}$  无需重复计算!

$$\begin{aligned}
 X_1(k) &= \sum_{r=0}^{N/2-1} x(2r) W_N^{2rk} = \sum_{r=0}^{N/2-1} x(2r) W_{N/2}^{rk} \\
 X_2(k) &= \sum_{r=0}^{N/2-1} x(2r+1) W_N^{2rk} = \sum_{r=0}^{N/2-1} x(2r+1) W_{N/2}^{rk}
 \end{aligned}$$

# 基于时间抽取 (DIT) 的快速傅里叶变换

$$\begin{aligned} X(k) &= \sum_{r=0}^{N/2-1} x(2r)W_N^{2rk} + \sum_{r=0}^{N/2-1} x(2r+1)W_N^{(2r+1)k} \\ &\Downarrow \\ X(k) &= X_1(k) + X_2(k)W_N^k \end{aligned} \quad \begin{aligned} X(k + \frac{N}{2}) &= \sum_{r=0}^{N/2-1} x(2r)W_N^{2r(k+N/2)} + \sum_{r=0}^{N/2-1} x(2r+1)W_N^{(2r+1)(k+N/2)} \\ &\Downarrow \\ X(k + \frac{N}{2}) &= X_1(k) - W_N^k X_2(k) \end{aligned}$$

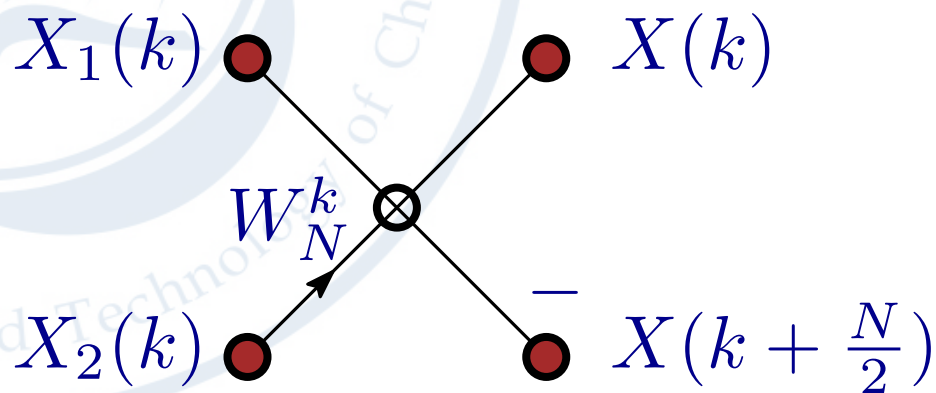
# 基于时间抽取 (DIT) 的快速傅里叶变换

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

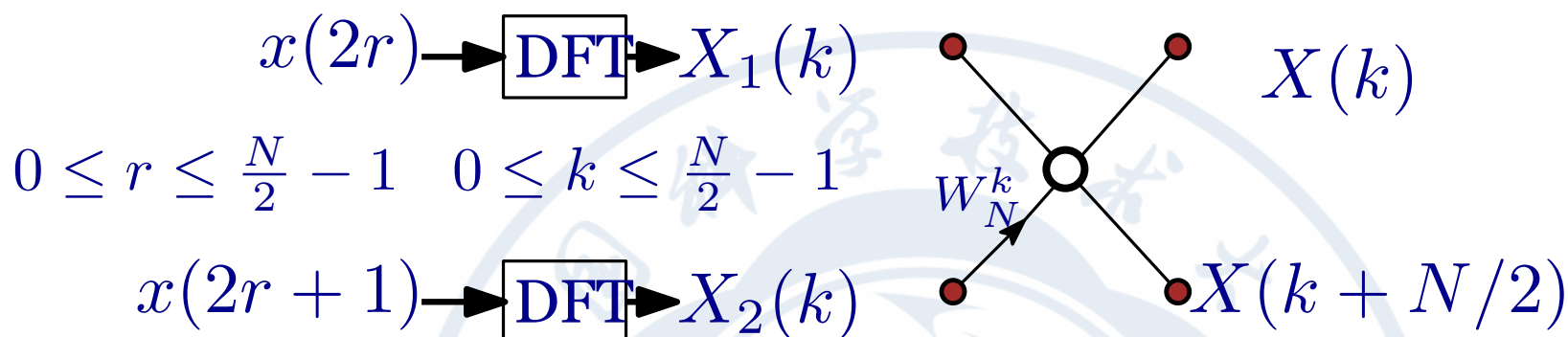
$$\begin{aligned}
 \Downarrow \quad X(k) &= X_1(k) + X_2(k)W_N^k \\
 \Downarrow \quad X(k + \frac{N}{2}) &= X_1(k) - W_N^k X_2(k)
 \end{aligned}$$

$$X_1(k) = \sum_{r=0}^{N/2-1} X(2r)W_{N/2}^{rk}$$

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



# 基于时间抽取 (DIT) 的快速傅里叶变换



- 1:  $x(n)$  分为偶序列  $x(2r)$  和奇下标序列  $x(2r+1)$ , 分别得到对应的 DFT 结果  $X_1(k)$  和  $X_2(k)$

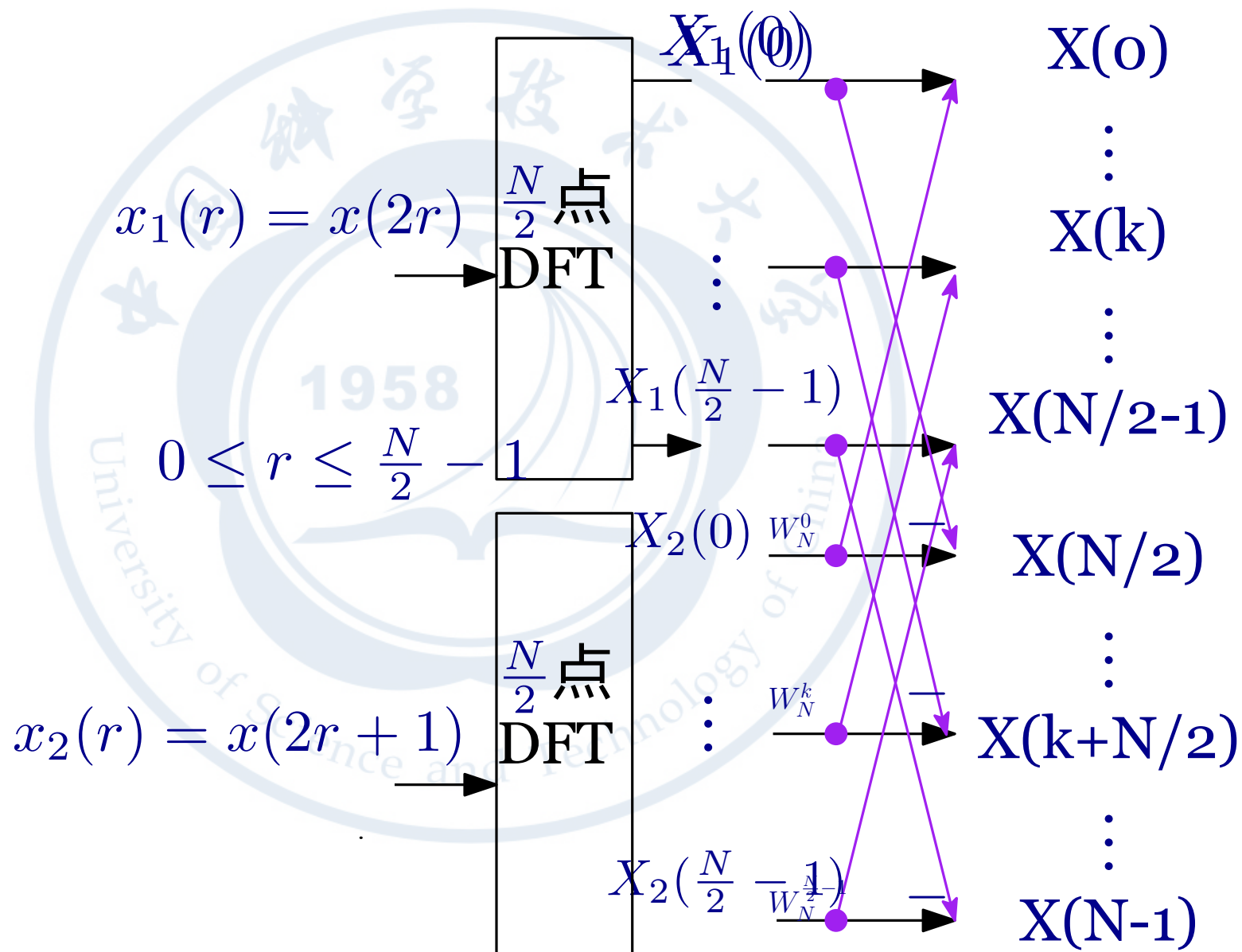
$N/2$  点 DFT 复数乘法次数  $N^2/4$ , 复数加法次数  $N(N-1)/4$

- 2.  $X_1(k)$  和  $X_2(k)$  合成  $X(k), X(k + N/2), 0 \leq k \leq N/2 - 1$

每个蝶形运算 1 次复数乘法, 2 次复数加法, 一共  $N/2$  个蝶形运算,  $N/2$  乘法,  $N$  次复数加法

# 基于时间抽取 DIT 的快速傅里叶变换

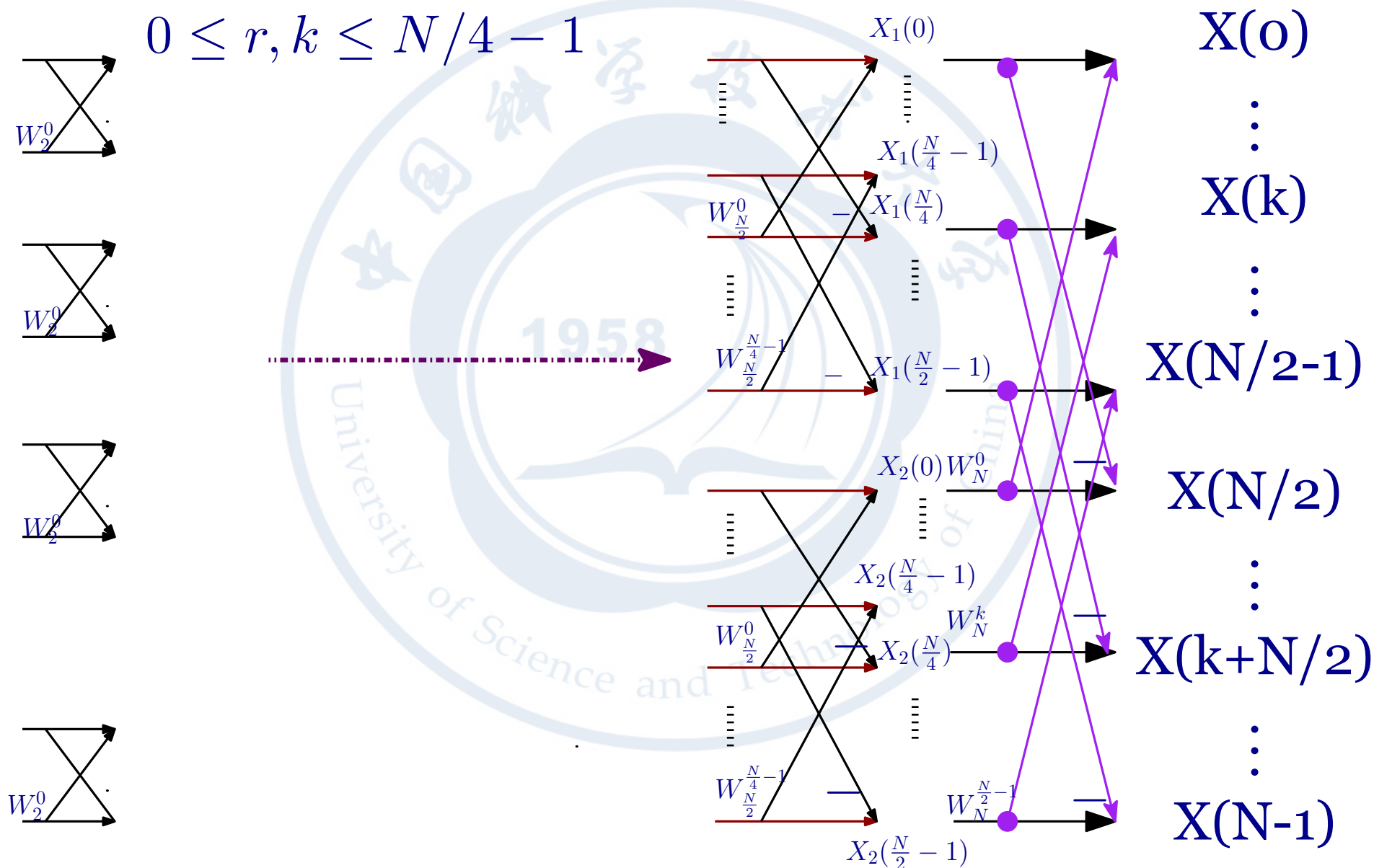
If  $N = 2^m$ , 则可以继续对  $x(2r), x(2r + 1)$  进行同样的操作



# 基于时间抽取 DIT 的快速傅里叶变换

If  $N = 2^m$ , 则可以继续对  $x(2r), x(2r + 1)$  进行同样的操作

$$0 \leq r, k \leq N/4 - 1$$



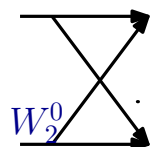


# 基于时间抽取 DIT 的快速傅里叶变换

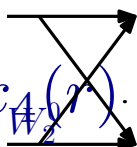
If  $N = 2^m$ , 则可以继续对  $x(2r), x(2r + 1)$  进行同样的操作

$$0 \leq r, k \leq N/4 - 1$$

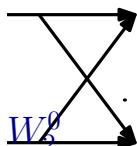
$$x_3(r) = x_1(2r) = x(4r)$$



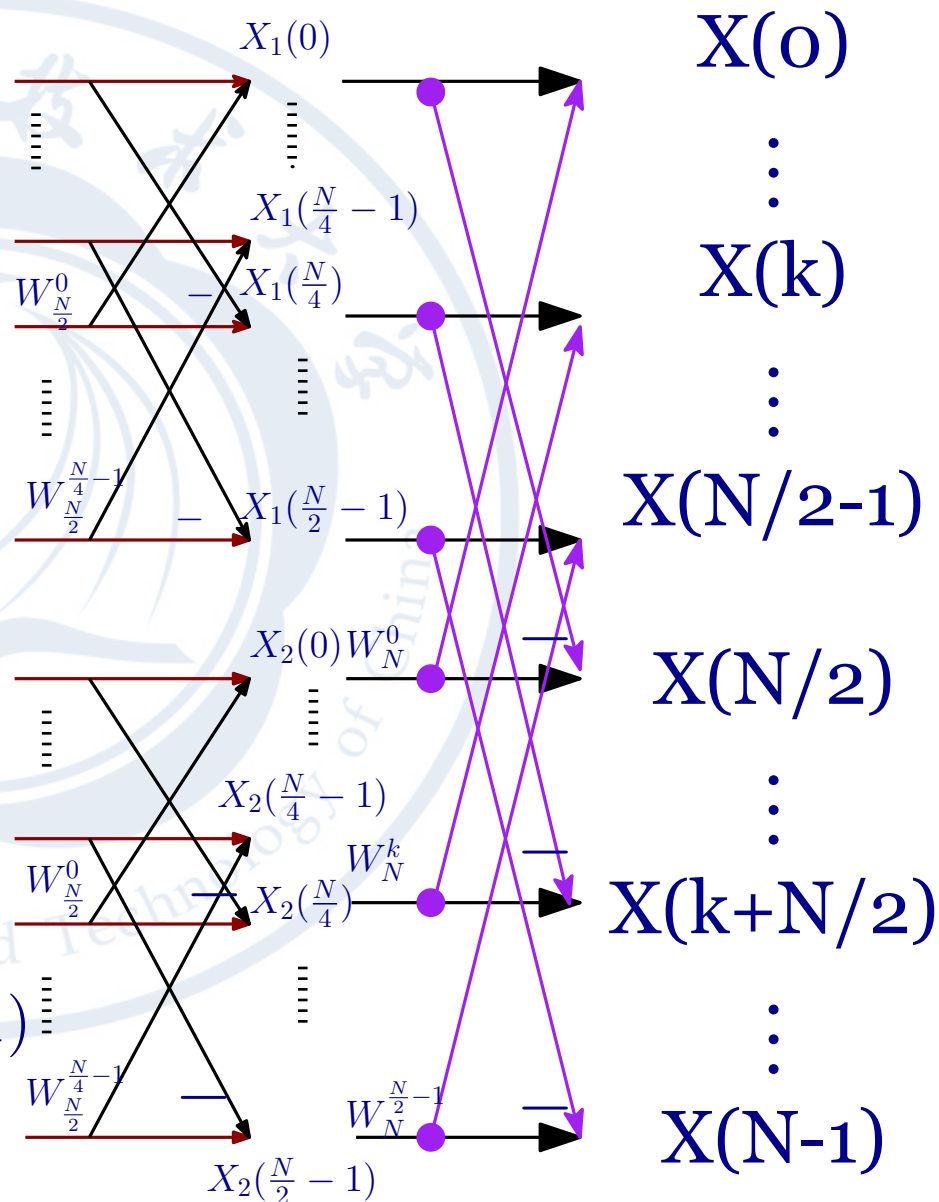
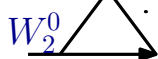
$$x_4(r) = x_1(2r + 1) = x(2(2r + 1))$$



$$x_5(r) = x_2(2r) = x(2(2r) + 1)$$



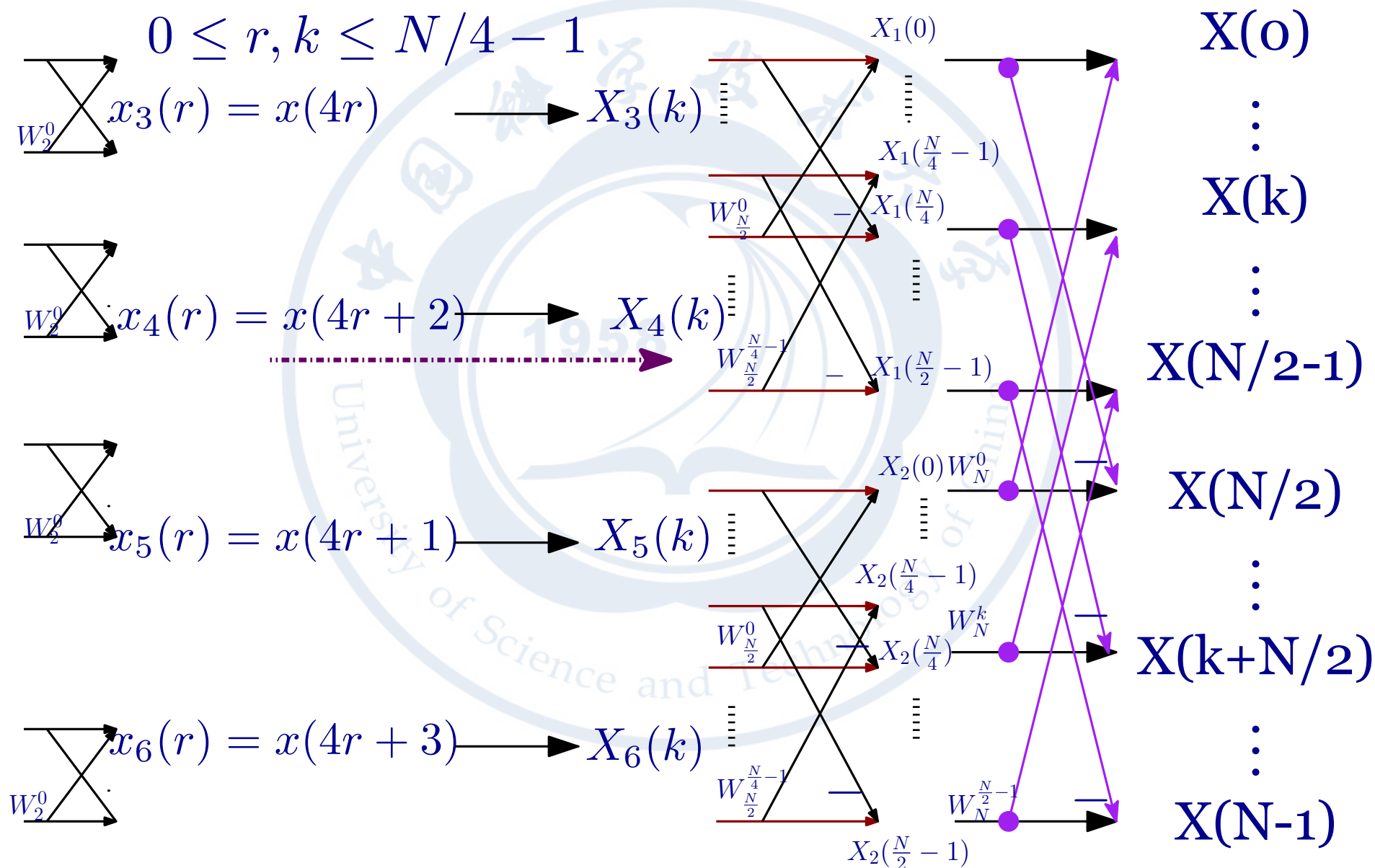
$$x_6(r) = x_2(2r + 1) = x(2(2r + 1) + 1)$$





# 基于时间抽取 DIT 的快速傅里叶变换

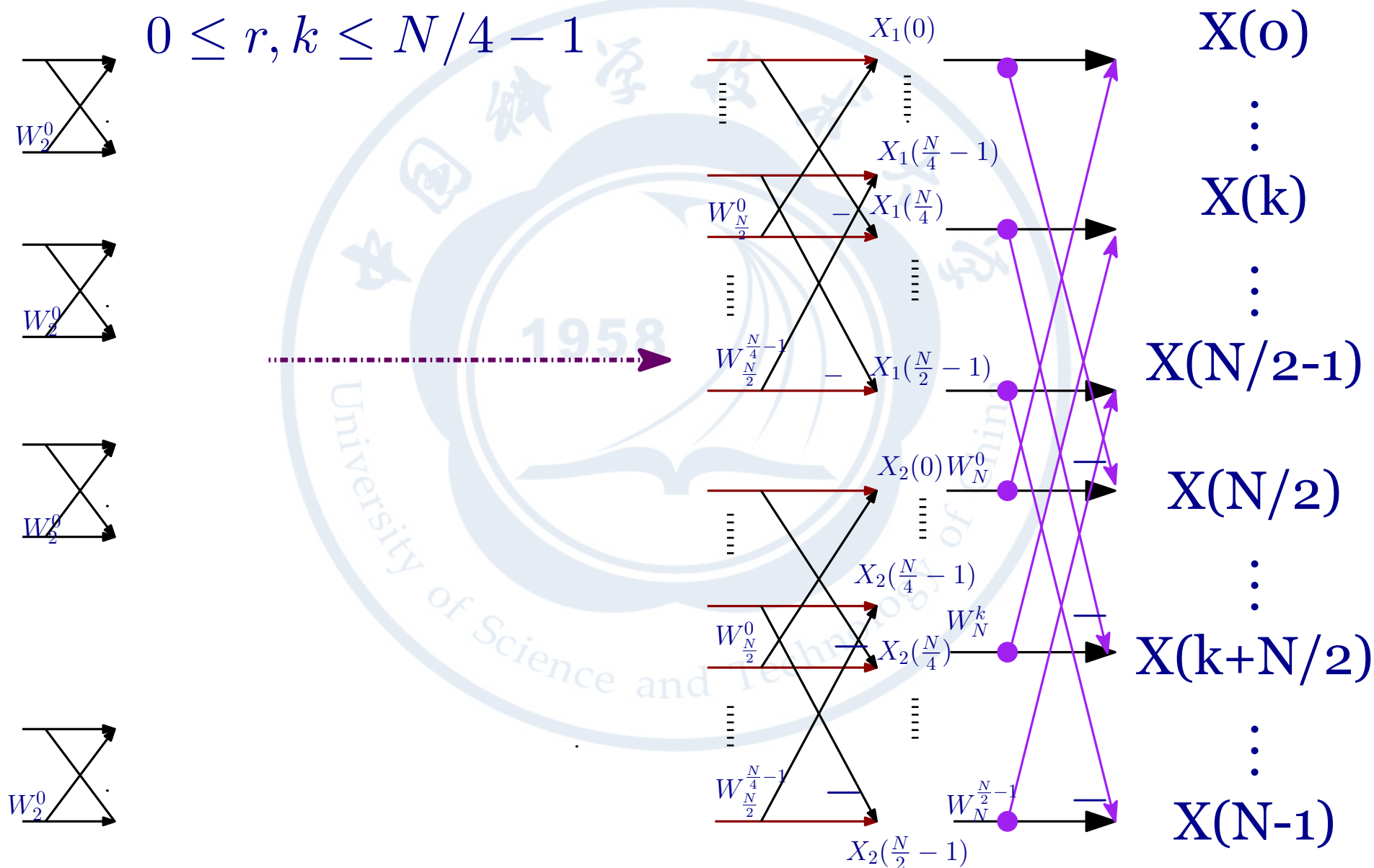
If  $N = 2^m$ , 则可以继续对  $x(2r), x(2r + 1)$  进行同样的操作



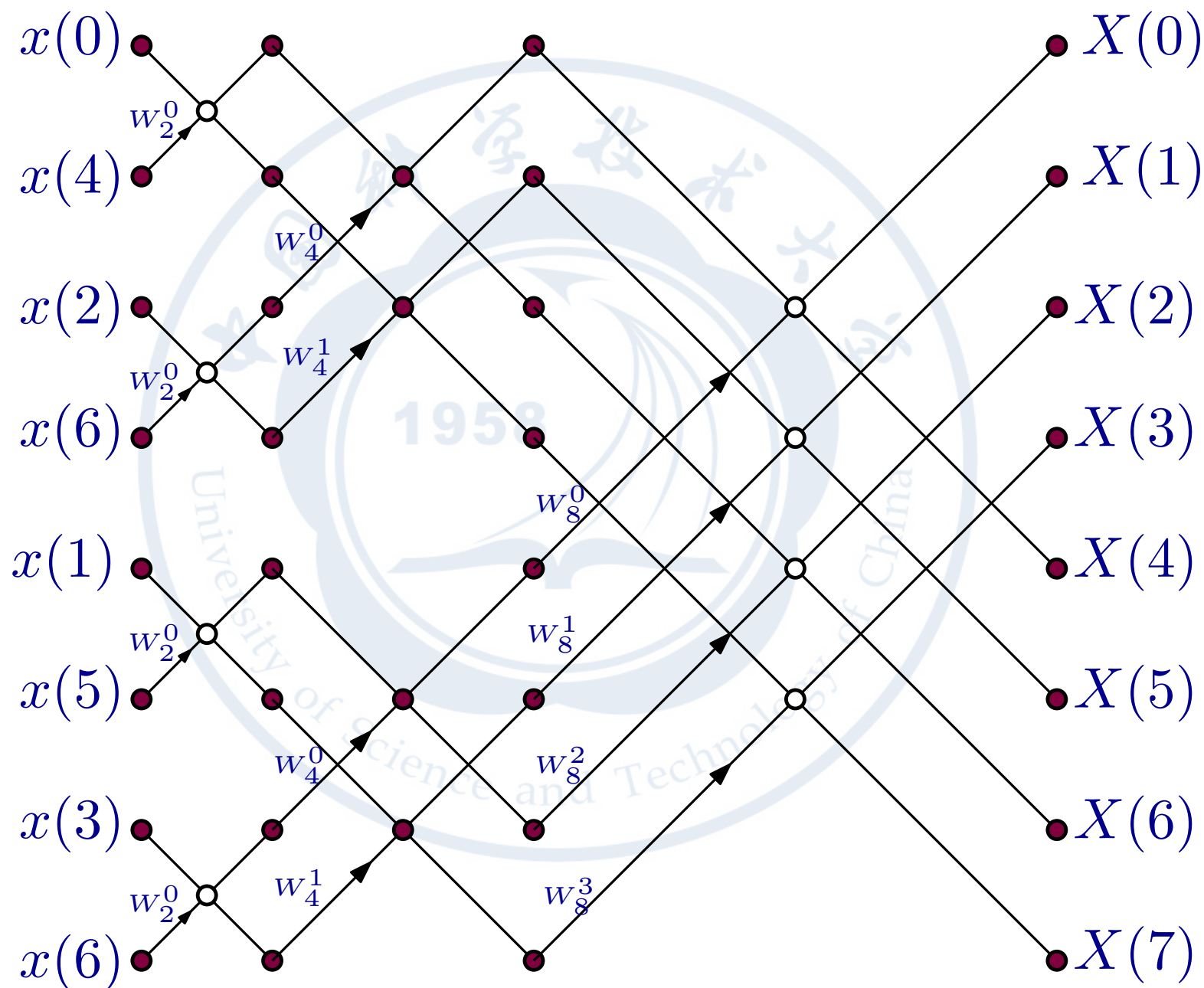
# 基于时间抽取 DIT 的快速傅里叶变换

If  $N = 2^m$ , 则可以继续对  $x(2r), x(2r + 1)$  进行同样的操作

$$0 \leq r, k \leq N/4 - 1$$



# 基于时间抽取 (DIT) 的快速傅里叶变换



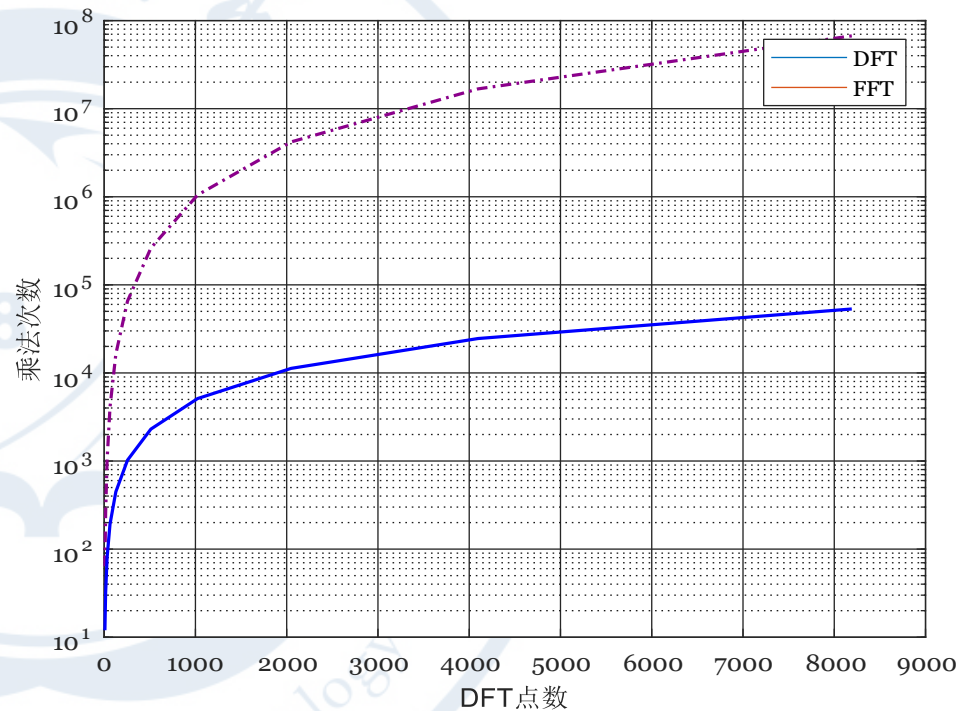
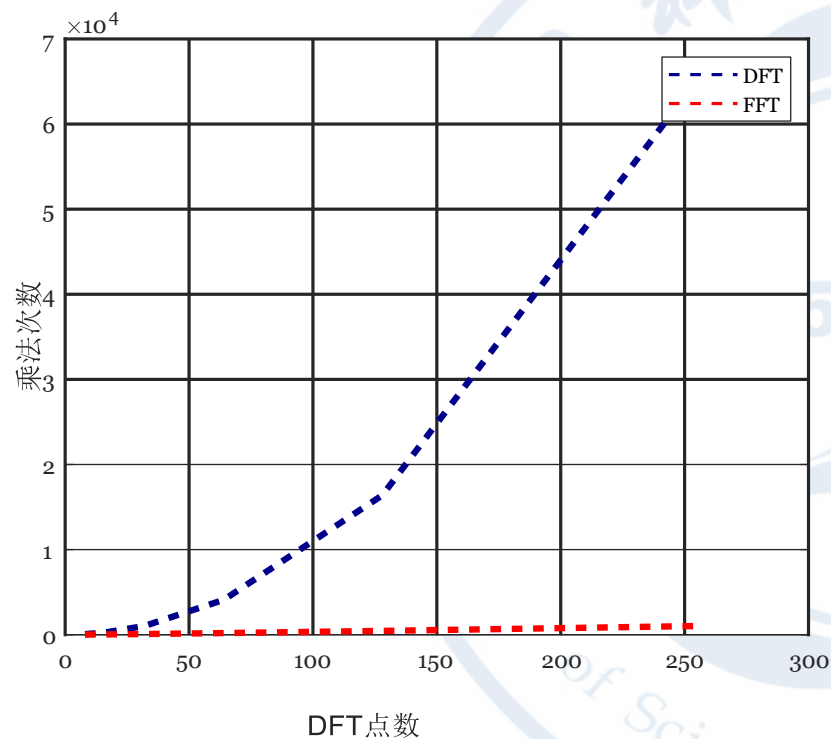
# 基于时间抽取的 (DIT) 快速算法

- 长度  $N = 2^m$  的 DFT, 可利用 DIT 反复分拆, 直到最后分拆为  $N/2$  个 2 点 DFT
- 第  $k$  次分拆时, 分拆为  $2^k$  个长度为  $2^{N-k}$  的 DFT
  - ◇ 每组 2 个长度为  $2^{N-k}$  的 DFT 合成  $2^{N-k+1}$  点 DFT, 一共  $2^{k-1}$  组
    - ◇ 每组合并涉及  $2^{N-k}$  个 DIT, 每个 DIT 1 次复乘, 2 次复加
    - ◇ 第  $k$  次拆分对应每组  $2^{N-k}$  个 DIT 的旋转因子为  $W_{2^{N-k+1}}^l$ , 其中  $0 \leq l \leq 2^{N-k} - 1$
    - ◇ 第  $k$  次运算每组  $2^{m-k}$  个 DIT, 一共  $2^{k-1}$  组
    - ◇ 乘法次数  $2^{m-k} \times 2^{k-1} = N/2$ , 加法次数  $N$
- 一共需要拆分  $m$  次, 需要乘法  $m2^{m-1} = N/2 \log_2 N$  次复数乘法,  $m2^m = N \log_2 N$  次复数加法

# 基于时间抽取 (DIT) 的DFT快速计算方法

## 复杂度分析

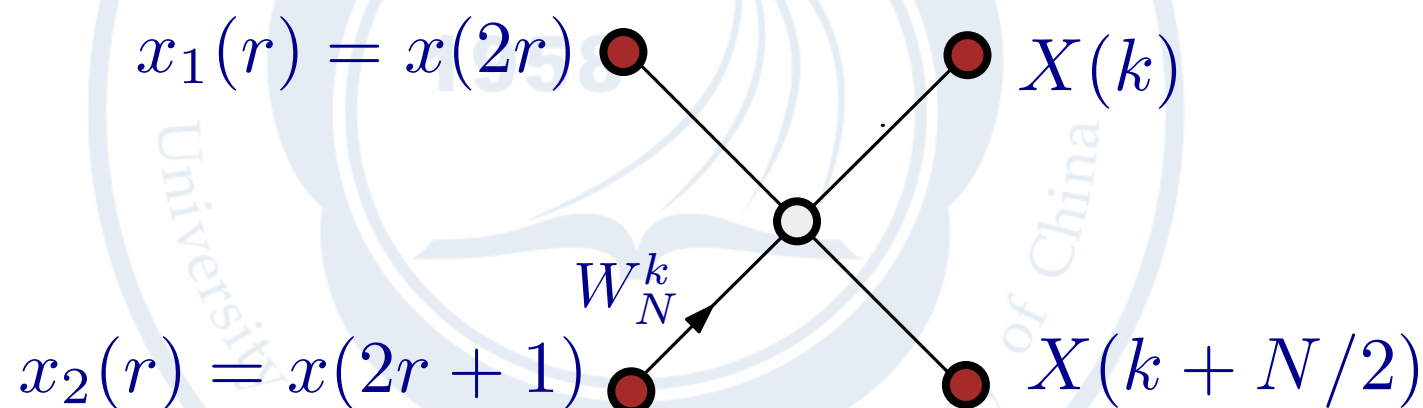
### 乘法次数对比



# 基于时间抽取 (DIT) 快速傅里叶变换

## 输入输出序列与真实序列位置对应

- 输出序列  $X(k), 0 \leq k \leq N-1$  顺序存放
- 输入序列  $x(n)$  运算关系



可以看出，时间信号的奇偶性决定  $x(n)$  处于  $X_1(r)$  还是  $X_2(r)$ ，考虑  $n$  的二进制表示  $b_0 b_1 \cdots b_{m-1}$ ,  $b_{m-1}$  决定，进一步  $b_{m-k}$  决定第  $m-k$  次分拆所处的位置， $b_{m-k} = 0$  偶数序列， $b_{m-k} = 1$  奇数序列



# 基于时间抽取 (DIT) 快速傅里叶变换

对  $x(n)$ , 序号  $n = (b_{m-1} \cdots b_0)_2$ , 第  $k$  次拆分奇下标序列和偶下标序列取决于从右数第  $k$  位  $b_{k-1} = 0$  or  $1$ , 则其在运算结构图中的序号对应为:

$$p_{n,\text{fft}} = \sum_{k=0}^{m-1} b_k 2^{m-k-1}$$

$x(n)$  按照时间先后位置是:

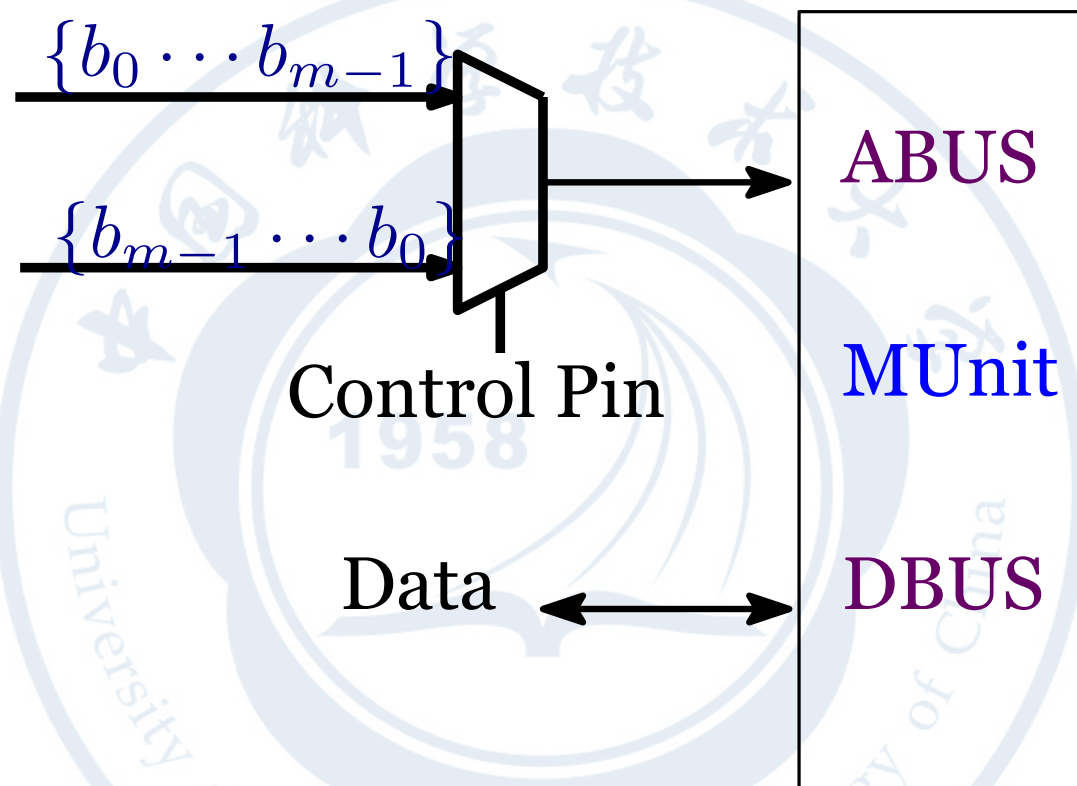
$$p_{n,t} = \sum_{k=0}^{m-1} b_{m-k} 2^{m-k-1}$$

按照时间抽取的  
fft,  $x(n)$  对应的位置是他的时间序号  
二进制序列反转对应的序号

如何快速的实现将按照时间先后的  $x(n)$  放入指定位置?



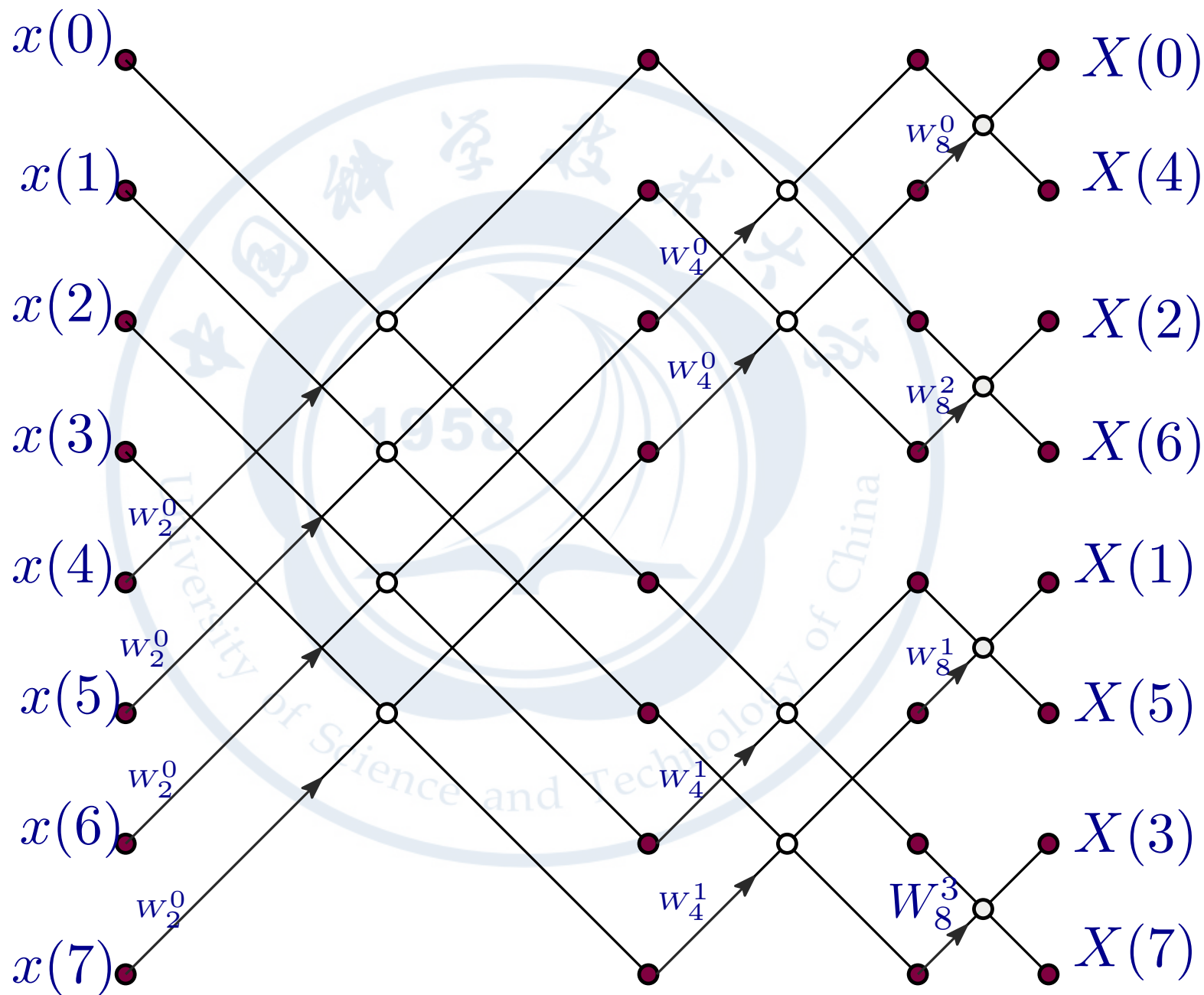
# 基于时间抽取 (DIT) 快速傅里叶变换



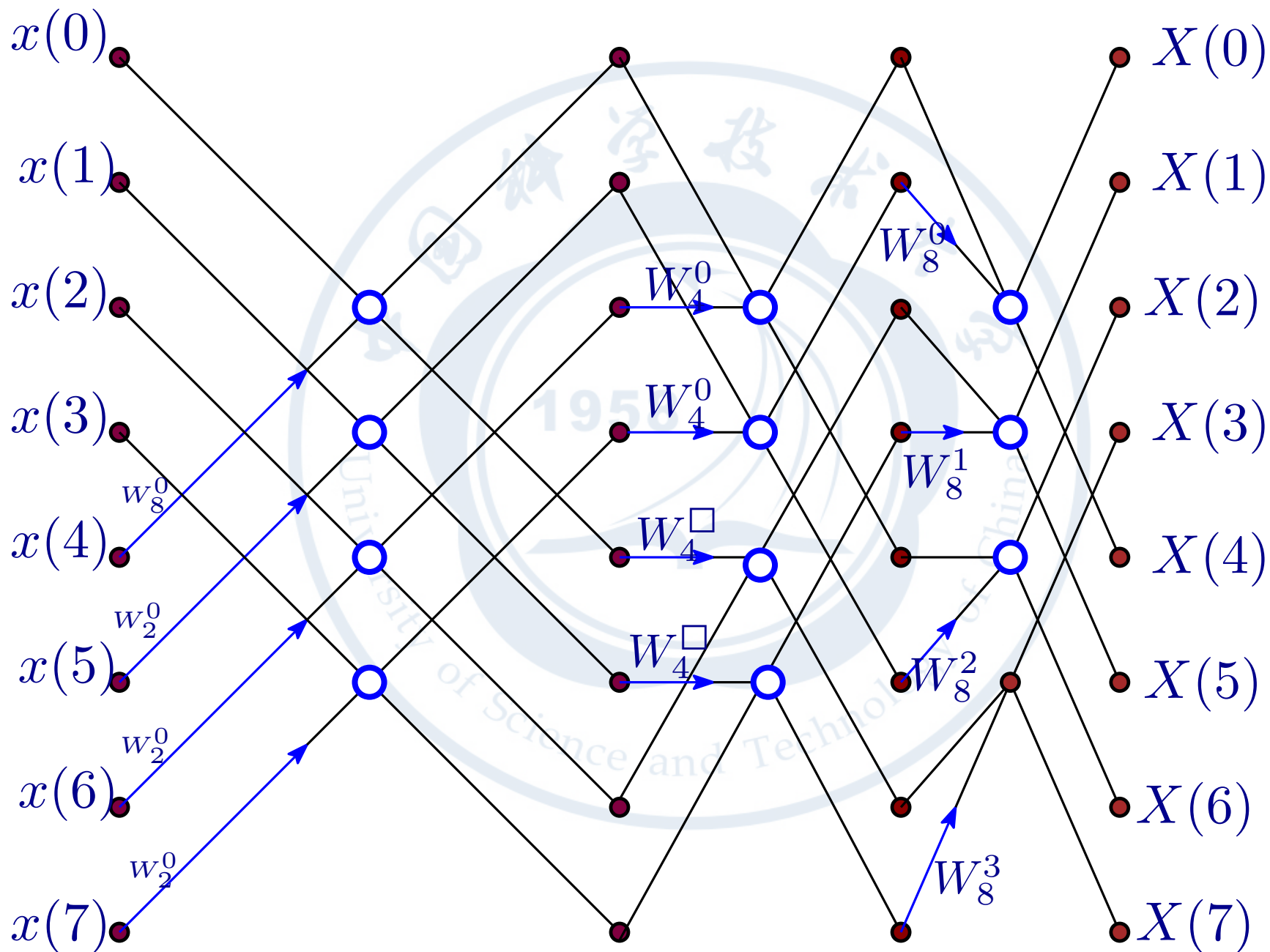
DSP Address Methods: Normal or FFT method

# 基于时间抽取 (DIT) 快速傅里叶变换

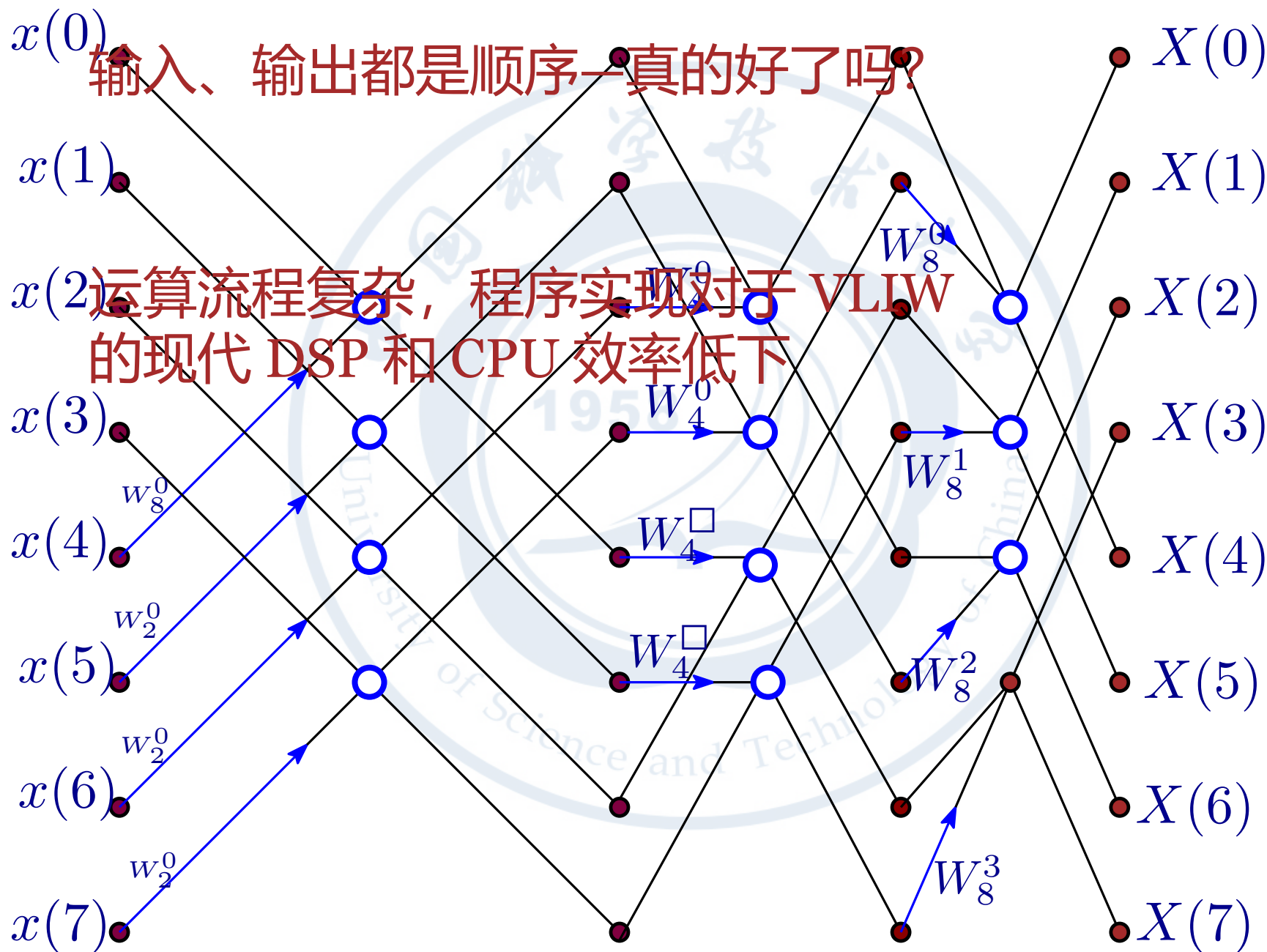
顺序



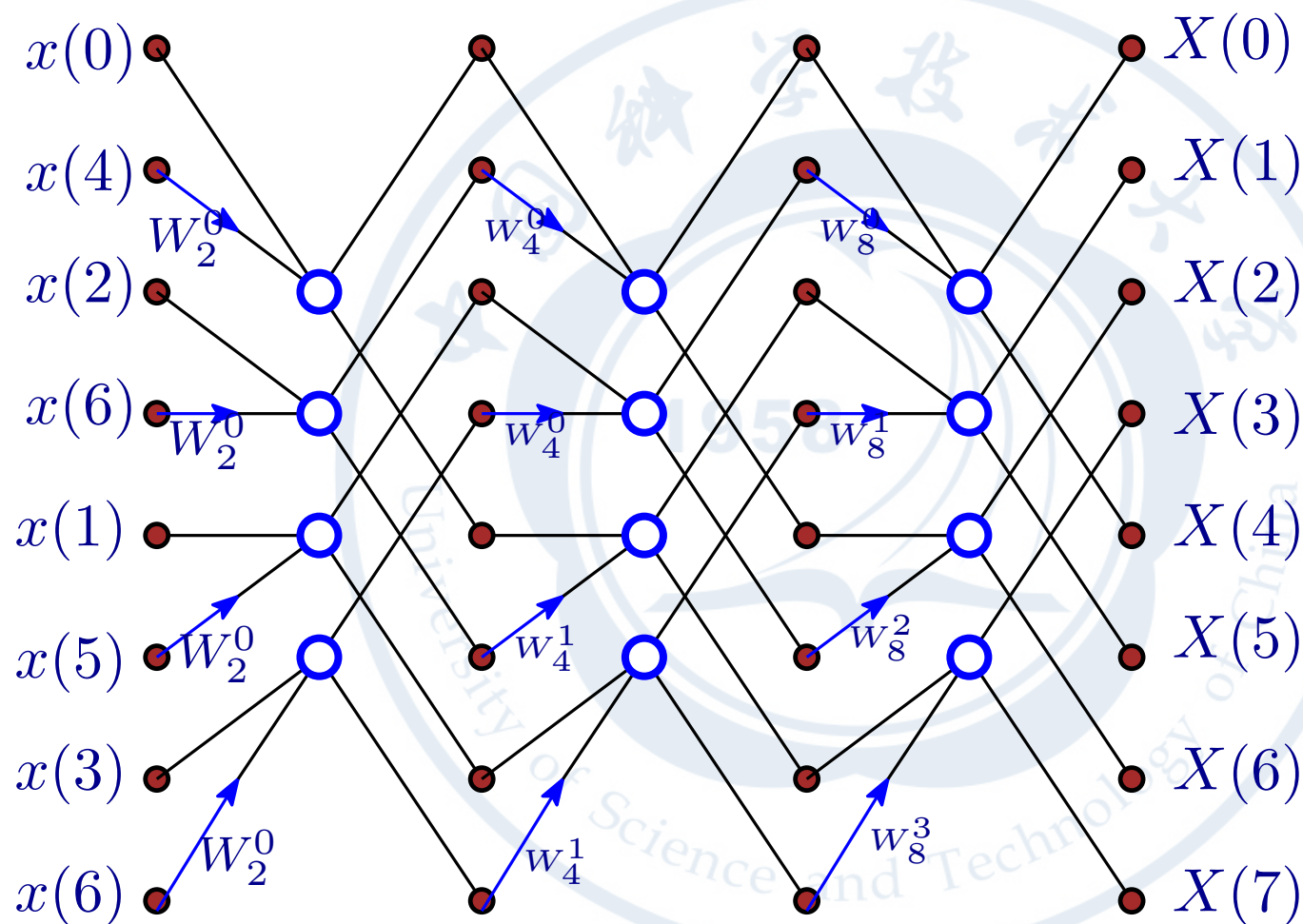
# 基于时间抽取 (DIT) 的快速傅里叶变换



# 基于时间抽取 (DIT) 的快速傅里叶变换



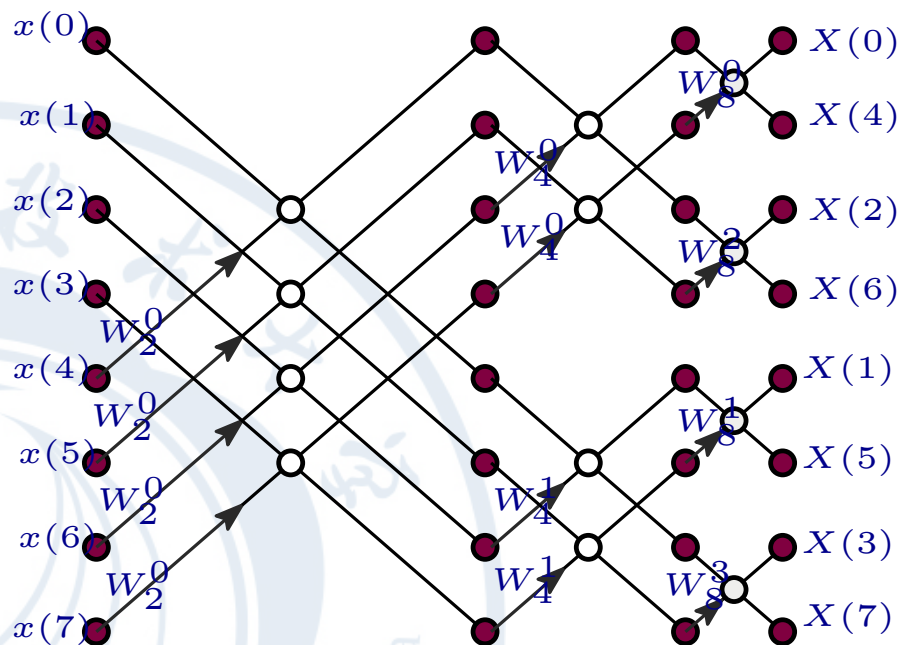
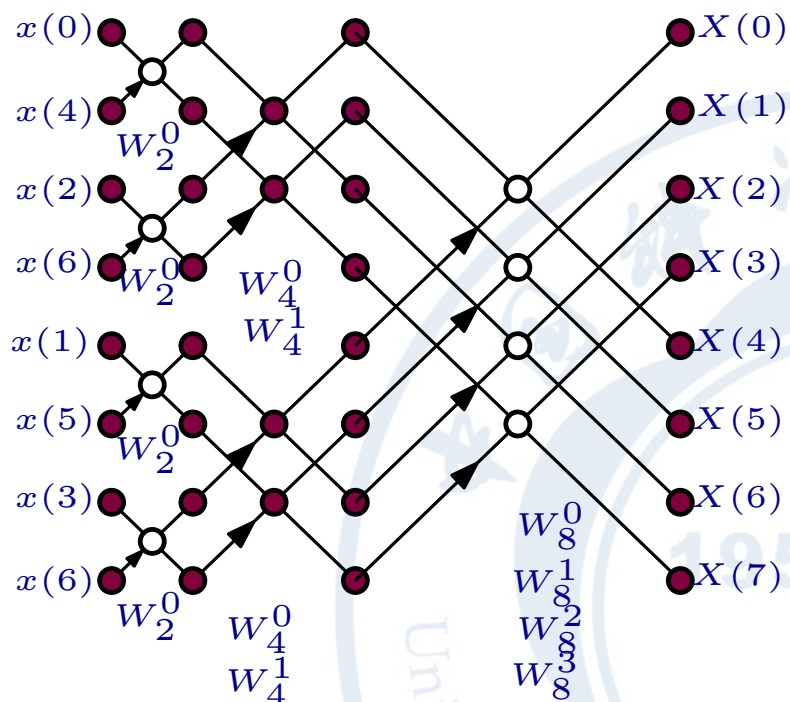
# 基于时间抽取 (DIT) 的快速傅里叶变换



考虑芯片流水线方式实现，可重复使用硬件资源，降低芯片面积和成本

恒定几何结构，可重复利用，输入反序，输出顺序

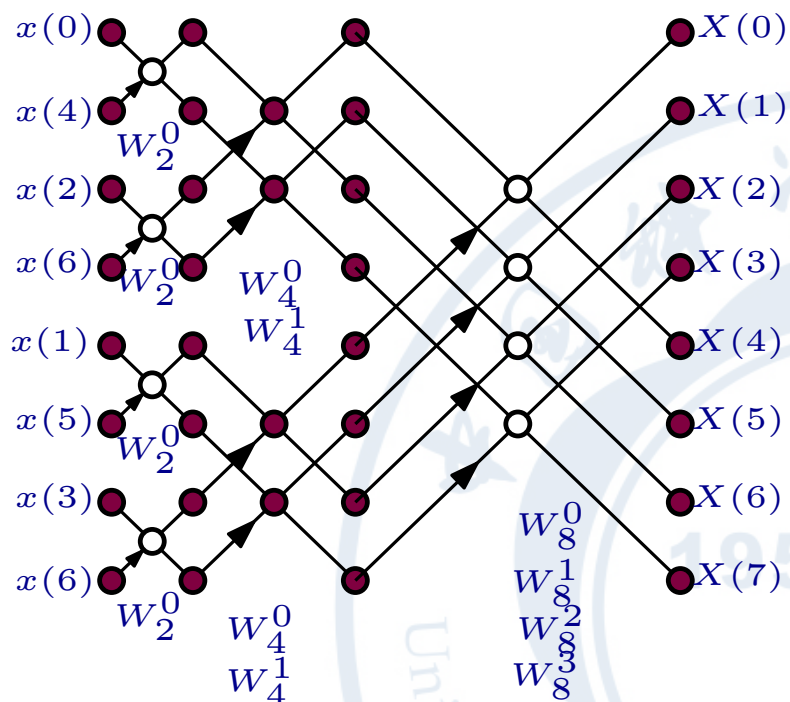
# FFT 中的原位运算



**原位运算：**基本运算单元输入数据和输出结果在同一个位置，无需额外分配暂存空间，即可进行下一步运算。

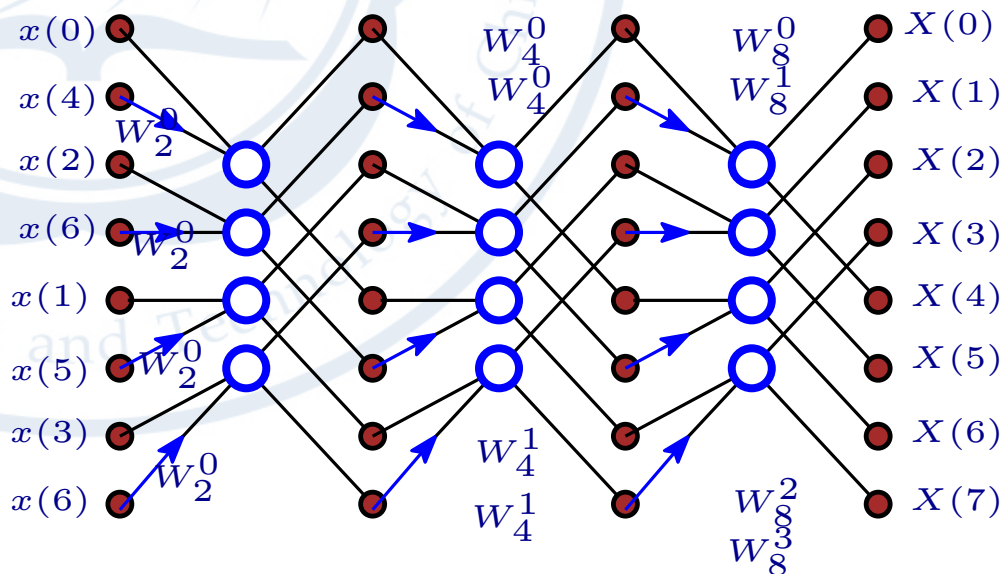
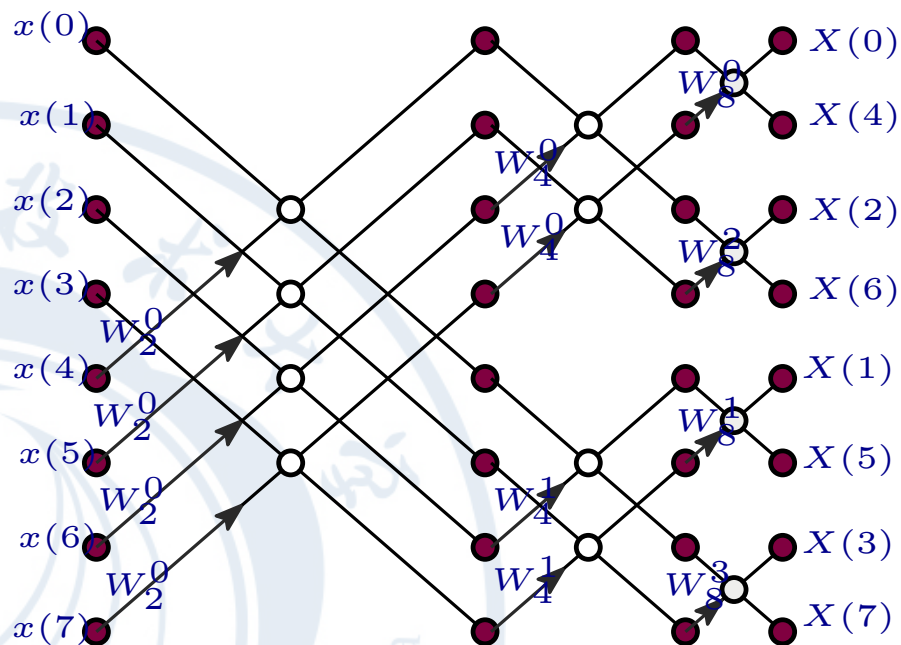


# FFT 中的原位运算



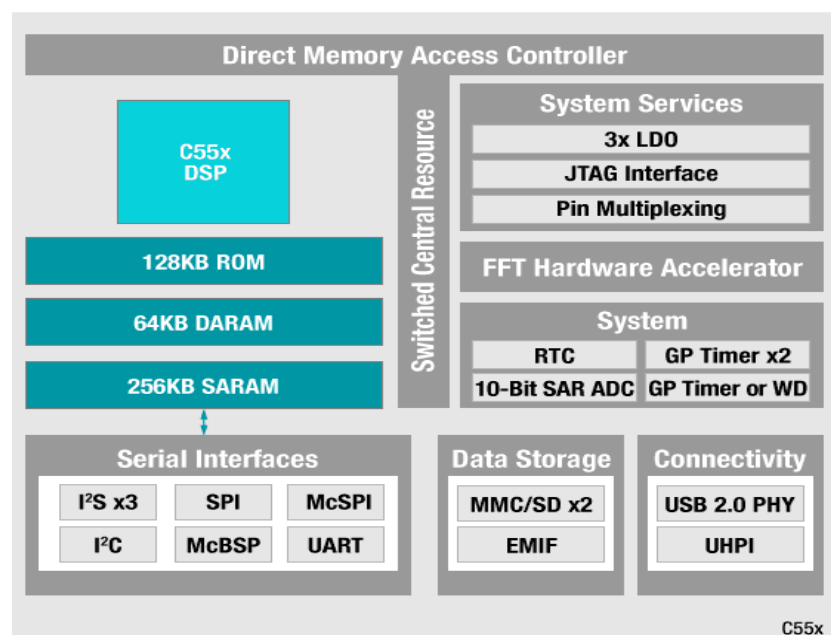
**原位运算：**基本运算单元输入数据和输出结果在同一个位置，无需额外分配暂存空间，即可进行下一步运算。

对于低成本嵌入式处理器，cache 资源与非常稀少，通常 xKBytes 这个数量级，能否原位运算可能非常重要，而且还少了数据移动所付出的系统开销





# 常见 DSP 的存储器资源



Type	C5401	C5407	C5410
Memory	8KB	32KB	64KB
Price(>1K Units)	3.3\$	11\$	42\$

定点处理器价格，在 90 年代量产

Type	C6742	C6746	C6748
Memory	64KB	256KB	256KB
Price(>1K Units)	5\$	8\$	9\$

浮点处理器价格，在 2010 前量产，主频有差异。

- 片上存储器才能做到单指令周期存储
- 多端口存储器并行读取是实现 FFT 快速并发的核心技术
- CPU 的流水线操作非常重要, 涉及到了大量跳转

# 基于频率抽取 (DIF) 的快速傅里叶变换

当  $n_1, n_2, k$  满足下属条件时 (与 DIT 对称) :

$$[k(n_1 - n_2)]\%N = 0 \Rightarrow W_N^{n_1 k} = W_N^{n_2 k}$$

$$[k(n_1 - n_2)]\%N = N/2 \Rightarrow W_N^{n_1 k} = -W_N^{n_2 k}$$

此时, 下述表达式成立:

$$W_N^{n_1 k} x(n_1) + W_N^{n_2 k} x(n_2) = W_N^{n_1 k} (x(n_1) \pm x(n_2))$$

原有 2 次复数乘法, 1 次复数加法, 降低为 1 次复数乘法, 1 次复数加法。

特别地, 当  $n_1 - n_2 = N/2$  时,  $\forall k \in N$  条件都满足时,  $k$  为偶数取正号,  $k$  为奇数时取负号

# 基于频率抽取 (DIF) 的快速傅里叶变换

考虑  $k = 2l, k = 2l + 1$  为 (频率域, 故称为**频域抽取**)

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

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

利用前面得到的结论:

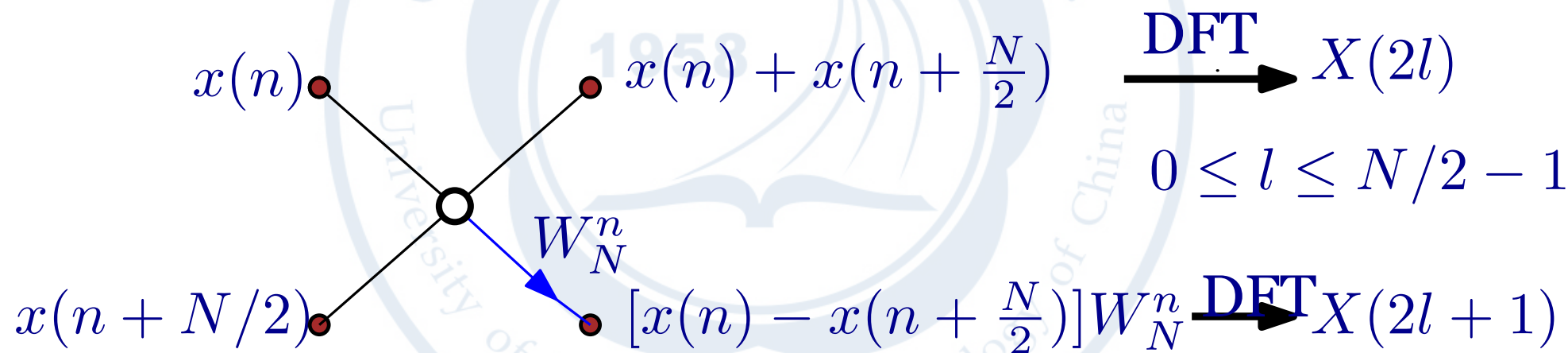
$$W_N^{n_1 k} x(n_1) + W_N^{n_2 k} x(n_2) = W_N^{n_1 k} (x(n_1) \pm x(n_2)) |_{n_1 - n_2 = N/2}$$

# 基于频率抽取 (DIF) 的快速傅里叶变换

于是:

$$X(2l) = \sum_{n=0}^{N/2-1} W_{N/2}^{ln} (x(n) + X(n + N/2))$$

$$X(2l + 1) = \sum_{n=0}^{N/2-1} W_{N/2}^{ln} \{ (x(n) - X(n + N/2)) W_N^n \}$$

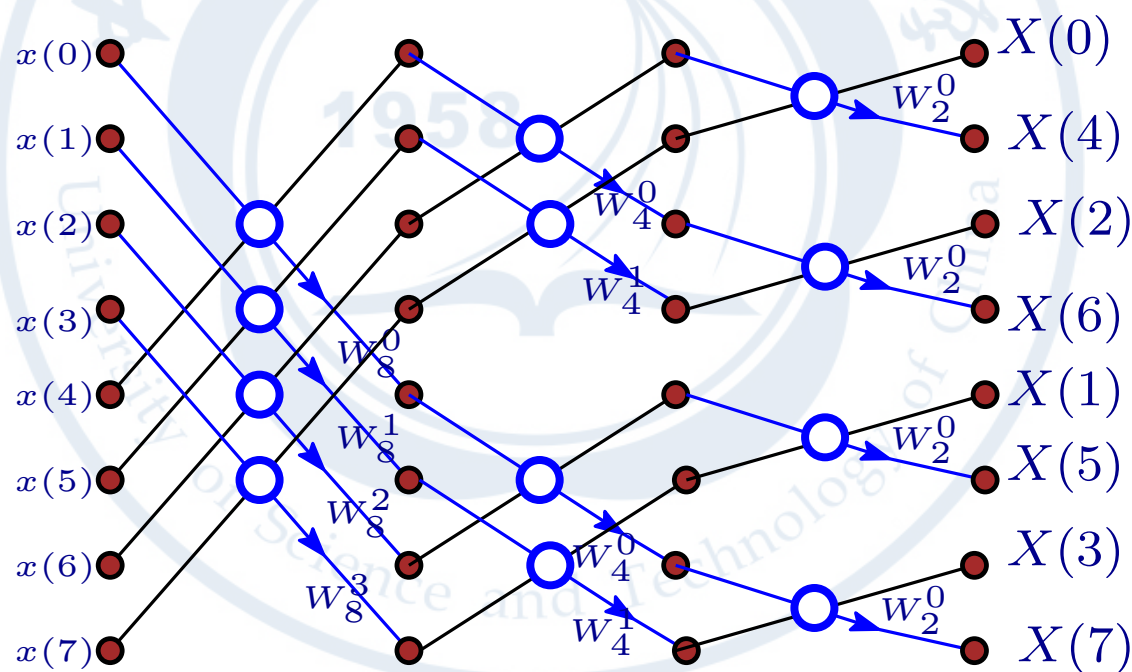


一个  $N$  点 DFT, 可以拆分为 2 个  $N/2$  点 DFT, 付出  $N/2$  个乘法,  $N$  个加法的系统开销

# 基于频率抽取 (DIF) 的快速傅里叶变换

思路:

对于  $N = 2^m$  形式的 DFT, 可以分拆为 2 个  $N/2$  点 DFT, 分别为对应偶数表述的离散傅里叶变换结果和奇数标号的离散傅里叶变换结构, 并可继续分解下去, 直到最后  $2^{m-1}$  个 2 点 DFT



# 基于频率抽取 (DIF) 的快速傅里叶变换

- DIT和 DIF运算流图上是互为逆向过程
- DIT所有实现方式均为转置均可更改为 DIF的实现方式
- DIF与 DIT有相同的实现复杂度

## 进一步可以思考的问题

- DIT与 DIF都是以 DFT 点数为 2 的倍数开展
- DIT,DIF条件:  $k(n_1 - n_2)\%N = 0$ ,  $n(k_1 - k_2)\%N = 0$

$$(n_1 - n_2) = N/2, 2|k, k_1 - k_2 = N/2, 2|n$$

$(n_1 - n_2)p\%N = 0, p|k$  或者  $(k_1 - k_2)p\%N = 0, p|n$ , 即  
 $N = p \times N/p$  如何计算?  $p, N/p$  均为整数



# $N$ 为复合数的快速傅里叶变换

**研究对象：**  $N = M \times L$ ,  $M, L$ 均为非 0, 1 的整数, 长度为  $N$  的序列的快速傅里叶变换方法。

$$(n_1 - n_2)L = lN, l \in N \Rightarrow W^{n_1 k} = W^{n_2 k}, k = mL$$

$$(k_1 - k_2)M = mN, m \in N \rightarrow W^{n k_1} = W^{n k_2}, n = lM$$

这意味着：

$$\text{if } n_1 = n_2 + lM, \text{ 有 } W^{n_1 k} = W^{n_2 k}, k = mL$$

$$\text{if } k_1 = k_2 + mL, \text{ 有 } W^{n k_1} = W^{n k_2}, n = lM$$



# $N$ 为复合数的快速傅里叶变换

**研究对象：**  $N = M \times L$ ,  $M, L$ 均为非 0, 1 的整数, 长度为  $N$  的序列的快速傅里叶变换方法。

$$(n_1 - n_2)L = lN, l \in N \Rightarrow W^{n_1 k} = W^{n_2 k}, k = mL$$
$$(k_1 - k_2)M = mN, m \in N \rightarrow W^{n k_1} = W^{n k_2}, n = lM$$

这意味着：

if  $n_1 = n_2 + lM$ , 有  $W^{n_1 k} = W^{n_2 k}, k = mL$   
if  $k_1 = k_2 + mL$ , 有  $W^{n k_1} = W^{n k_2}, n = lM$

时间序列分为  $L$  段, 频率域每隔  $L$  点旋转因子重复; 频率域分为  $M$  段, 时间序列每隔  $M$  点旋转因子重复。

思路：我们是否可以利用上述两个思路进行算法简化？

$$n = Mn_1 + n_0, k = Lk_1 + k_0$$
$$0 \leq k_1, n_0 \leq M - 1, 0 \leq n_1, k_0 \leq L_1$$

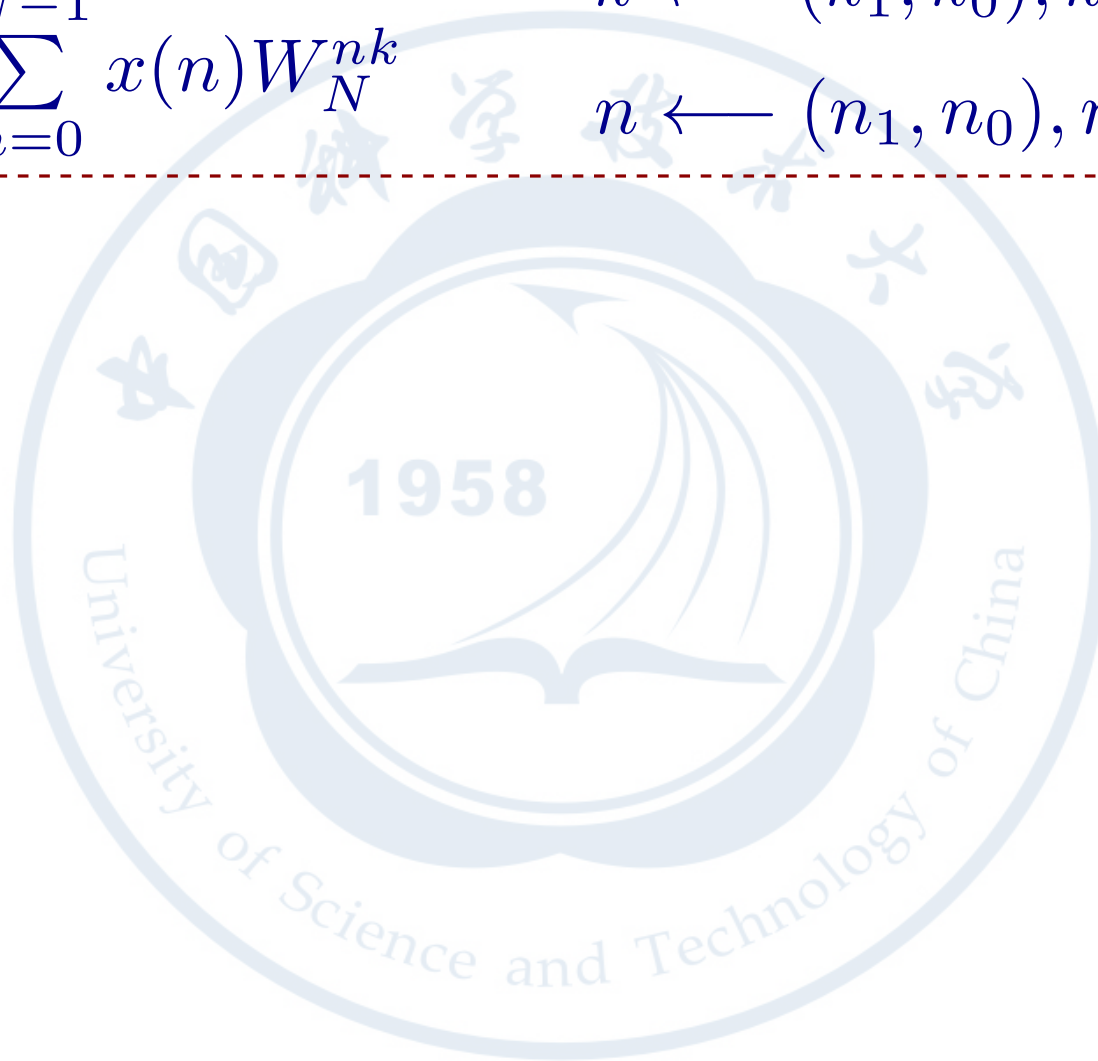
# $N$ 为复合数的快速傅里叶变换

$n \backslash n_0$	0	$\dots$	$n_0$	$\dots$	$M - 1$
$n_1$					
0	$x(0)$	$\dots$	$x(n_0)$	$\dots$	$x(M - 1)$
1	$x(1M + 0)$	$\dots$	$x(M + n_0)$	$\dots$	$x(2M - 1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$n_1$	$x(n_1M + 0)$	$\dots$	$x(n_1M + n_0)$	$\dots$	$x(n_1M - 1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$L - 1$	$x((L - 1)M + 0)$	$\dots$	$x((L - 1)M + n_0)$	$\dots$	$x(LM - 1)$

$$(n_1, n_0) \rightarrow x(n_1M + n_0)$$

# $N$ 为复合数的快速傅立叶变换

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad \begin{array}{l} k \leftarrow (k_1, k_0), k = k_1 L + k_0 \\ n \leftarrow (n_1, n_0), n = n_1 M + n_0 \end{array}$$



# $N$ 为复合数的快速傅立叶变换

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad \begin{array}{l} k \leftarrow (k_1, k_0), k = k_1 L + k_0 \\ n \leftarrow (n_1, n_0), n = n_1 M + n_0 \end{array}$$

$$X(Lk_1 + k_0) = \sum_{n_0=0}^{M-1} \sum_{n_1=0}^{L-1} x(n_1 M + n_0) W_N^{(n_1 M + n_0)(Lk_1 + k_0)}$$

# $N$ 为复合数的快速傅立叶变换

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad \begin{array}{l} k \leftarrow (k_1, k_0), k = k_1 L + k_0 \\ n \leftarrow (n_1, n_0), n = n_1 M + n_0 \end{array}$$

$$X(Lk_1 + k_0) = \sum_{n_0=0}^{M-1} \sum_{n_1=0}^{L-1} x(n_1 M + n_0) W_N^{(n_1 M + n_0)(Lk_1 + k_0)}$$

$$W_N^{(n_1 M + n_0)(Lk_1 + k_0)} = W_N^{n_0 k_0} W_M^{n_0 k_1} W_L^{n_1 k_0} W_N^{n_1 k_1 L M}$$

# $N$ 为复合数的快速傅立叶变换

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad \begin{array}{l} k \leftarrow (k_1, k_0), k = k_1 L + k_0 \\ n \leftarrow (n_1, n_0), n = n_1 M + n_0 \end{array}$$

$$X(Lk_1 + k_0) = \sum_{n_0=0}^{M-1} \sum_{n_1=0}^{L-1} x(n_1 M + n_0) W_N^{(n_1 M + n_0)(Lk_1 + k_0)}$$

$$W_N^{(n_1 M + n_0)(Lk_1 + k_0)} = W_N^{n_0 k_0} W_M^{n_0 k_1} W_L^{n_1 k_0} W_N^{n_1 k_1 L M}$$

$$X(Lk_1 + k_0) = \sum_{n_0=0}^{M-1} \left\{ W_N^{n_0 k_0} \left[ \sum_{n_1=0}^{L-1} x(Mn_1 + n_0) W_L^{n_1 k_0} \right] \right\} W_M^{n_0 k_1}$$



# $N$ 为复合数的快速傅立叶变换

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad \begin{aligned} k &\leftarrow (k_1, k_0), k = k_1 L + k_0 \\ n &\leftarrow (n_1, n_0), n = n_1 M + n_0 \end{aligned}$$

$$X(Lk_1 + k_0) = \sum_{n_0=0}^{M-1} \sum_{n_1=0}^{L-1} x(n_1 M + n_0) W_N^{(n_1 M + n_0)(Lk_1 + k_0)}$$

$$W_N^{(n_1 M + n_0)(Lk_1 + k_0)} = W_N^{n_0 k_0} W_M^{n_0 k_1} W_L^{n_1 k_0} W_N^{n_1 k_1 L M}$$

$$X(Lk_1 + k_0) = \sum_{n_0=0}^{M-1} \left\{ W_N^{n_0 k_0} \left[ \sum_{n_1=0}^{L-1} x(Mn_1 + n_0) W_L^{n_1 k_0} \right] \right\} W_M^{n_0 k_1}$$

定义:  $x(n_1, n_0) = x(Mn_1 + n_0)$ ,  $X(k_1, k_0) = X(Lk_1 + k_0)$

$$X(k_1, k_0) = \sum_{n_0=0}^{M-1} \left\{ W_N^{n_0 k_0} \left[ \sum_{n_1=0}^{L-1} x(n_1, n_0) W_L^{n_1 k_0} \right] \right\} W_M^{n_0 k_1}$$

# $N$ 为复合数的快速傅里叶变换

$$X(k_1, k_0) = \sum_{n_0=0}^{M-1} \left\{ W_N^{n_0 k_0} \underbrace{\left[ \sum_{n_1=0}^{L-1} x(n_1, n_0) W_L^{n_1 k_0} \right]}_{X(k_0, n_0)} \right\} W_M^{n_0 k_1}$$

**Step 1:** 对于  $x(n_1, n_0)$ , 对于任何一个给定的  $n_0$ , 对应一个  $L$  点的 DFT, 注意到  $0 \leq n_0 \leq M-1$ , 一共有  $M$  组  $L$  点 DFT

# $N$ 为复合数的快速傅里叶变换

$$X(k_1, k_0) = \sum_{n_0=0}^{M-1} \left\{ W_N^{n_0 k_0} \underbrace{\left[ \sum_{n_1=0}^{L-1} x(n_1, n_0) W_L^{n_1 k_0} \right]}_{X(k_0, n_0)} \right\} W_M^{n_0 k_1}$$

**Step 1:** 对于  $x(n_1, n_0)$ , 对于任何一个给定的  $n_0$ , 对应一个  $L$  点的 DFT, 注意到  $0 \leq n_0 \leq M-1$ , 一共有  $M$  组  $L$  点 DFT

$x(n) \backslash n_0$	0	...	$n_0$	...	$M-1$
$n_1$					
0	$x(0)$	...	$x(n_0)$	...	$x(M-1)$
1	$x(M+0)$	...	$x(M+n_0)$	...	$x(2M-1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n_1$	$x(n_1 M + 0)$	...	$x(n_1 M + n_0)$	...	$x(n_1 M - 1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$L-1$	$x((L-1)M + 0)$	...	$x((L-1)M + n_0)$	...	$x(LM-1)$

# $N$ 为复合数的快速傅里叶变换

$$X(k_1, k_0) = \sum_{n_0=0}^{M-1} \left\{ W_N^{n_0 k_0} \underbrace{\left[ \sum_{n_1=0}^{L-1} x(n_1, n_0) W_L^{n_1 k_0} \right]}_{X(k_0, n_0)} \right\} W_M^{n_0 k_1}$$

**Step 1:** 对于  $x(n_1, n_0)$ , 对于任何一个给定的  $n_0$ , 对应一个  $L$  点的 DFT, 注意到  $0 \leq n_0 \leq M-1$ , 一共有  $M$  组  $L$  点 DFT

$n \backslash n_0$	0	...	$n_0$	...	$M-1$
$n_1$					
0	$x(0, 0)$	...	$x(0, n_0)$	...	$x(0, M-1)$
1	$x(1, 0)$	...	$x(1, n_0)$	...	$x(1, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$n_1$	$x(n_1, 0)$	...	$x(n_1, n_0)$	...	$x(n_1, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$L-1$	$x((L-1), 0)$	...	$x((L-1), n_0)$	...	$x(L-1, M-1)$

# $N$ 为复合数的快速傅里叶变换

$$X(k_1, k_0) = \sum_{n_0=0}^{M-1} \left\{ W_N^{n_0 k_0} \underbrace{\left[ \sum_{n_1=0}^{L-1} x(n_1, n_0) W_L^{n_1 k_0} \right]}_{X(k_0, n_0)} \right\} W_M^{n_0 k_1}$$

**Step 1:** 对于  $x(n_1, n_0)$ , 对于任何一个给定的  $n_0$ , 对应一个  $L$  点的 DFT, 注意到  $0 \leq n_0 \leq M-1$ , 一共有  $M$  组  $L$  点 DFT

$n \backslash n_0$	0	...	$n_0$	...	$M-1$
$n_1$					
0	$x(0, 0)$	...	$x(0, n_0)$	...	$x(0, M-1)$
1	$x(1, 0)$	...	$x(1, n_0)$	...	$x(1, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$n_1$	$x(n_1, 0)$	...	$x(n_1, n_0)$	...	$x(n_1, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$L-1$	$x((L-1), 0)$	...	$x((L-1), n_0)$	...	$x(L-1, M-1)$

# $N$ 为复合数的快速傅里叶变换

$$X(k_1, k_0) = \sum_{n_0=0}^{M-1} \left\{ W_N^{n_0 k_0} \underbrace{\left[ \sum_{n_1=0}^{L-1} x(n_1, n_0) W_L^{n_1 k_0} \right]}_{X(k_0, n_0)} \right\} W_M^{n_0 k_1}$$

**Step 1:** 对于  $x(n_1, n_0)$ , 对于任何一个给定的  $n_0$ , 对应一个  $L$  点的 DFT, 注意到  $0 \leq n_0 \leq M-1$ , 一共有  $M$  组  $L$  点 DFT

$n \backslash n_0$	0	...	$n_0$	...	$M-1$
$k_0$					
0	$X(0, 0)$	...	$X(0, n_0)$	...	$X(0, M-1)$
1	$X(1, 0)$	...	$X(1, n_0)$	...	$X(1, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$n_1$	$X(n_1, 0)$	...	$X(n_1, n_0)$	...	$X(n_1, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$L-1$	$X((L-1), 0)$	...	$X((L-1), n_0)$	...	$X(L-1, M-1)$



# $N$ 为复合数快速傅里叶变换

$$X(k_1, k_0) = \sum_{n_0=0}^{M-1} \underbrace{\left\{ W_N^{n_0 k_0} X(k_0, n_0) \right\}}_{X_1(k_0, n_0)} W_M^{n_0 k_1}$$

**Step2:**对 Step 1 每一步结果  $X(k_0, n_0)$  乘上旋转因子  $W_N^{n_0 k_0}$  得到  $X_1(k_0, n_0)$

# $N$ 为复合数快速傅里叶变换

$$X(k_1, k_0) = \sum_{n_0=0}^{M-1} \underbrace{\left\{ W_N^{n_0 k_0} X(k_0, n_0) \right\}}_{X_1(k_0, n_0)} W_M^{n_0 k_1}$$

**Step2:**对 Step 1 每一步结果  $X(k_0, n_0)$  乘上旋转因子  $W_N^{n_0 k_0}$  得到  $X_1(k_0, n_0)$

$n \backslash n_0$ $k_0$	0	...	$n_0$	...	$M-1$
0	$X_1(0, 0)$	...	$X_1(0, n_0)$	...	$X_1(0, M-1)$
1	$X_1(1, 0)$	...	$X_1(1, n_0)$	...	$X_1(1, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$k_0$	$X_1(k_0, 0)$	...	$X_1(k_0, n_0)$	...	$X_1(k_0, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$L-1$	$X_1((L-1), 0)$	...	$X_1((L-1), n_0)$	...	$X_1((L-1), M-1)$

# $N$ 为复合数快速傅里叶变换

$$X_2(k_0, k_1) = \sum_{n_0=0}^{M-1} X_1(k_0, n_0) W_M^{n_0 k_1} \leftarrow \text{以 } k_0 \text{ 为参数的 } L \text{ 个长度为 } M \text{ 的 DFT}$$

$n \backslash n_0$					
$k_0 \backslash$	0	...	$n_0$	...	$M-1$
0	$X_1(0, 0)$	...	$X_1(0, n_0)$	...	$X_1(0, M-1)$
1	$X_1(1, 0)$	...	$X_1(1, n_0)$	...	$X_1(1, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$k_0$	$X_1(k_0, 0)$	...	$X_1(k_0, n_0)$	...	$X_1(k_0, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$L-1$	$X_1((L-1), 0)$	...	$X_1((L-1), n_0)$	...	$X_1((L-1), M-1)$

# $N$ 为复合数快速傅里叶变换

$$X_2(k_0, k_1) = \sum_{n_0=0}^{M-1} X_1(k_0, n_0) W_M^{n_0 k_1} \leftarrow \text{以 } k_0 \text{ 为参数的 } L \text{ 个长度为 } M \text{ 的 DFT}$$

**Step3:** 对于  $X_1(k_0, n_0), 0 \leq n_0 \leq M-1, 0 \leq k_0 \leq L-1$ , 可以看成是  $L$  个  $M$  点序列  $X_1(k_0, \cdot)$  的 DFT,  $0 \leq k_0 \leq L-1$  是序列的序号

$n \backslash n_0$ $k_0$	0	...	$n_0$	...	$M-1$
0	$X_1(0, 0)$	...	$X_1(0, n_0)$	...	$X_1(0, M-1)$
1	$X_1(1, 0)$	...	$X_1(1, n_0)$	...	$X_1(1, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$k_0$	$X_1(k_0, 0)$	...	$X_1(k_0, n_0)$	...	$X_1(k_0, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$L-1$	$X_1((L-1), 0)$	...	$X_1((L-1), n_0)$	...	$X_1((L-1), M-1)$

# $N$ 为复合数快速傅里叶变换

$$X_2(k_0, k_1) = \sum_{n_0=0}^{M-1} X_1(k_0, n_0) W_M^{n_0 k_1} \leftarrow \text{以 } k_0 \text{ 为参数的 } L \text{ 个长度为 } M \text{ 的 DFT}$$

**Step3:** 对于  $X_1(k_0, n_0)$ ,  $0 \leq n_0 \leq M-1$ ,  $0 \leq k_0 \leq L-1$ , 可以看成是  $L$  个  $M$  点序列  $X_1(k_0, \cdot)$  的 DFT,  $0 \leq k_0 \leq L-1$  是序列的序号

$n \backslash n_0$ $k_0$	0	...	$n_0$	...	$M-1$
0	$X_1(0, 0)$	...	$X_1(0, n_0)$	...	$X_1(0, M-1)$
1	$X_1(1, 0)$	...	$X_1(1, n_0)$	...	$X_1(1, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$k_0$	$X_1(k_0, 0)$	...	$X_1(k_0, n_0)$	...	$X_1(k_0, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$L-1$	$X_1((L-1), 0)$	...	$X_1((L-1), n_0)$	...	$X_1((L-1), M-1)$

# $N$ 为复合数快速傅里叶变换

$$X_2(k_0, k_1) = \sum_{n_0=0}^{M-1} X_1(k_0, n_0) W_M^{n_0 k_1} \leftarrow \text{以 } k_0 \text{ 为参数的 } L \text{ 个长度为 } M \text{ 的 DFT}$$

**Step3:**, 对于  $X_1(k_0, n_0)$ ,  $0 \leq n_0 \leq M_1, 0 \leq k_0 \leq L-1$ , 可以看成是  $L$  个  $M$  点序列  $X_1(k_0, \cdot)$  的 DFT,  $0 \leq k_0 \leq L-1$  是序列的序号

$X_2(k_1, k_0) \begin{matrix} \backslash k_1 \\ k_0 \end{matrix}$	0	...	$k_1$	...	$M-1$
0	$X_2(0, 0)$	...	$X_2(0, k_1)$	...	$X_1(0, M-1)$
1	$X_2(1, 0)$	...	$X_2(1, k_1)$	...	$X_2(1, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$k_0$	$X_2(k_0, 0)$	...	$X_2(k_0, k_2)$	...	$X_2(k_0, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$L-1$	$X_2((L-1), 0)$	...	$X_2((L-1), k_1)$	...	$X_2(L-1, M-1)$



# $N$ 为复合数快速傅里叶变换

$$X_2(k_0, k_1) = \sum_{n_0=0}^{M-1} X_1(k_0, n_0) W_M^{n_0 k_1} \leftarrow \text{以 } k_0 \text{ 为参数的 } L \text{ 个长度为 } M \text{ 的 DFT}$$

**Step3:**, 对于  $X_1(k_0, n_0)$ ,  $0 \leq n_0 \leq M_1, 0 \leq k_0 \leq L-1$ , 可以看成是  $L$  个  $M$  点序列  $X_1(k_0, \cdot)$  的 DFT,  $0 \leq k_0 \leq L-1$  是序列的序号

$X_2(k_1, k_0) \backslash k_1$	0	...	$k_1$	...	$M-1$
$k_0$					
0	$X_2(0, 0)$	...	$X_2(0, k_1)$	...	$X_1(0, M-1)$
1	$X_2(1, 0)$	...	$X_2(1, k_1)$	...	$X_2(1, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$k_0$	$X_2(k_0, 0)$	...	$X_2(k_0, k_2)$	...	$X_2(k_0, M-1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$L-1$	$X_2((L-1), 0)$	...	$X_2((L-1), k_1)$	...	$X_2(L-1, M-1)$

$$X(Lk_1 + k_0) = X_2(k_0, k_1)$$

# $N$ 为复合数快速傅里叶变换

$$X_2(k_0, k_1) = \sum_{n_0=0}^{M-1} X_1(k_0, n_0) W_M^{n_0 k_1} \leftarrow \text{以 } k_0 \text{ 为参数的 } L \text{ 个长度为 } M \text{ 的 DFT}$$

**Step3:**, 对于  $X_1(k_0, n_0), 0 \leq n_0 \leq M_1, 0 \leq k_0 \leq L-1$ , 可以看成是  $L$  个  $M$  点序列  $X_1(k_0, \cdot)$  的 DFT,  $0 \leq k_0 \leq L-1$  是序列的序号

$X(k) \backslash k_1$	0	...	$k_1$	...	$M-1$
$k_0$					
0	$X(0)$	...	$X(k_1 L)$	...	$X((M-1)L)$
1	$X(1)$	...	$X(k_1 L + 1)$	...	$X((M-1)L + 1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$k_0$	$X(k_0)$	...	$X(Lk_1 + k_0)$	...	$X((M-1)L + k_0)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$L-1$	$X(L-1)$	...	$X((Lk_1 + L-1))$	...	$X(LM-1)$

$$X(Lk_1 + k_0) = X_2(k_0, k_1)$$

# $N$ 为复合数的快速傅里叶变换

$$x(n) = x(n_1, n_0), n = n_1M + n_0, 0 \leq n_0 < M, 0 \leq n_1 < L$$

$$X_2(k_1, k_0) = \sum_{n_0=0}^{M-1} \underbrace{\left\{ W_N^{n_0 k_0} \underbrace{\left[ \sum_{n_1=0}^{L-1} x(n_1, n_0) W_L^{n_1 k_0} \right]}_{X(k_0, n_0)} \right\}}_{X_1(k_0, n_0)} W_M^{n_0 k_1}$$

**Step 1:** 对于  $x(n_1, n_0)$ , 对于任何一个给定的  $n_0$ , 对应一个  $L$  点的 DFT, 注意到  $0 \leq n_0 < M - 1$ , 一共有  $M$  组  $L$  点 DFT

**Step 2:** 对 Step 1 每一步结果  $X(k_0, n_0)$  乘上旋转因子  $W_N^{n_0 k_0}$  得到  $X_1(k_0, n_0)$

**Step 3:** 对于  $X_1(k_0, n_0)$ ,  $0 \leq n_0 \leq M_1, 0 \leq k_0 \leq L - 1$ , 可以看成是  $L$  个  $M$  点序列  $X_1(k_0, \cdot)$  的 DFT,  $0 \leq k_0 < L - 1$  是序列的序号

$$X(k) = X_2(k_0, k_1), k = Lk_1 + k_0, 0 \leq k_0 < L, 0 \leq k_1 < M$$

# $N$ 为复合数的快速傅里叶变换

计算复杂度 ( $0 \leq n_0, k_1 < M, 0 \leq n_1, k_0 < L$ ):

步骤	主要完成事项	计算复杂度
<b>S</b>	数据排序 $x(n_1, n_0) = x(n_1 M + n_0)$	—
<b>1</b>	$\forall n_0, X(k_0, n_0) = \text{DFT}(x(n_1, n_0))$	$LM^2 \times$
<b>2</b>	旋转 $X_1(k_0, n_0) = W_N^{n_0 k_0} X(k_0, n_0)$	$LM$
<b>3</b>	$\forall k_0, X_2(k_0, k_1) = \text{DFT}(X_1(k_0, n_0))$	$ML^2 \times$
<b>E</b>	数据排序 $X(Lk_1 + k_0) = X_2(k_0, k_1)$	—
合计		$LM^2 + L^2 M + LM$

**例 1:**  $N = 600, L = 20, M = 30$ ,  
本算法  $18000 + 12000 + 600 = 30600$ , 原始算法 360000

**例:**  $N = 600, L = 15, M = 40$ ,  
本算法  $9000 + 24000 + 600 = 33600$ , 原始算法 360000

# $N$ 为复合数的快速傅里叶算法

## $N = r^M$ 时可达到的快速傅里叶算法复杂度

- $f(m)$ 表示  $N = r^m$ 的乘法运算复杂度，特别  $m = 1, f(1) = r^2$
- $N = r^{m+1} = r^m \times r$ 计算复杂度：  
$$f(m+1) = rf(m) + r^m f(1) + r^{m+1}$$
- 进一步分解，一直分解，最后全为  $r$ 点 dft，全过程需要  $r^m$ 个  $r$ 点 dft， $m$ 次  $r^{m+1}$ 点乘法，于是总的运算复杂度为：  
$$f(m+1) = mr^m(f(r) + r)$$

思考：  $f(r)$ 对系统运算复杂度的影响，特别考虑  $r = 2, r = 3, r = 4$ 的运算复杂度对比



# N为复合数的快速傅立叶算法

## 算法复杂度分析:

$$f(m) = rf(m-1) + r^{m-1}f(1) + r^m = rf(m-1) + r^m(r+1)$$

进一步分解, 我们可以得到:

$$f(m-1) = rf(m-2) + r^{m-1} \times (r+1)$$

$$\longrightarrow f(m) = r^2 f(m-2) + 2r^m \times (r+1)$$

.....

$$\longrightarrow f(m) = r^k f(m-k) + kr^m \times (r+1) = mr^m(r+1)$$

## 物理解释:

每一次缩短分解, 基本操作都是  $r$  点 DFT 和旋转因子, 总长度为  $r^m$ , 所以  $r$  点 DFT 一共  $r^{m-1}$ , 所有的点都需要旋转所以旋转因子是  $r^m$  个, 这个分解每次缩短为  $1/r$ ,  $m$  次结束。所以总运算量为  $m(r^{m-1}f(1) + r^m) = mr^m(r+1)$

进一步思考: 对于 2, 4 点 DFT, 没有乘法, 因此运算量应该是  $m2^m$  和  $m4^m$ 。



# $N$ 为复合数的快速傅里叶变换

$N = 4^m$ 的运算次数为： $m4^m$ ，原因 4点 dft 无需任何乘法运算

$N = 2^m$ 是运算次数为： $m2^m$ ，因为 2点无需乘法运算

修正：

每次旋转因子相乘  $r^k \times r$  中第一行  $r^k$  个点无需乘法，所以可以进一步降低为  $mr^m / (r - 1)$ .

结论：基-2，基-4的快速傅里叶变化算法，基于其他的整数基傅里叶变化算法极少

<https://www.ti.com/lit/an/spra554b/spra554b.pdf>

<https://www.ti.com/lit/an/spra297/spra297.pdf>

# 关于 FFT 的讨论

为何基-2, 4 可以省运算量?



# 关于 FFT 的讨论

为何基-2, 4 可以省运算量?

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

$W_N^{k+N/2} = -W_N^k$ ! 这是能够将不符合简化运算的列也转化为简化运算的关键!  $x(2r+1)W_N^{(2r+1)k}$  和  $x(2r+1)W_N^{(2r+1)(k+N/2)}$  无需重复计算!

# 关于 FFT 的讨论

## 为何基-2, 4 可以省运算量?

$$\begin{aligned} X(k) &= \sum_{r=0}^{N/4-1} x(4r) W_N^{4rk} \\ &+ \sum_{r=0}^{N/4-1} x(4r+1) W_N^{(4r+1)k} \\ &+ \sum_{r=0}^{N/4-1} x(4r+2) W_N^{(4r+2)k} \\ &+ \sum_{r=0}^{N/4-1} x(4r+3) W_N^{(4r+3)k} \end{aligned}$$

$$\begin{aligned} X(k+N/4) &= \sum_{r=0}^{N/4-1} x(4r) W_N^{4rk} \\ &+ \sum_{r=0}^{N/4-1} x(4r+1) W_N^{(4r+1)(k+N/4)} \\ &+ \sum_{r=0}^{N/4-1} x(4r+2) W_N^{(4r+2)(k+N/4)} \\ &+ \sum_{r=0}^{N/4-1} x(4r+3) W_N^{(4r+3)(k+N/4)} \end{aligned}$$

$$W_N^{m(k+lN/4)} = j^{lm} W_N^{mk}, j = \sqrt{-1}, m, l \in \mathbb{Z} \text{ 简化运算的关键!}$$

$$x(4r+m) W_N^{(4r+m)k} \text{ 和 } x(4r+m) W_N^{(4r+m)(k+lN/4)} \text{ 无需重复计算!}$$

$$\begin{aligned} X(k+N/4) &= \sum_{r=0}^{N/4-1} x(4r) W_N^{4r(k+2N/4)} \\ &+ \sum_{r=0}^{N/4-1} x(4r+1) W_N^{(4r+1)(k+2N/4)} \\ &+ \sum_{r=0}^{N/4-1} x(4r+2) W_N^{(4r+2)(k+2N/4)} \\ &+ \sum_{r=0}^{N/4-1} x(4r+3) W_N^{(4r+3)(k+2N/4)} \end{aligned}$$

$$\begin{aligned} X(k+N/4) &= \sum_{r=0}^{N/4-1} x(4r) W_N^{4r(k+3N/4)} \\ &+ \sum_{r=0}^{N/4-1} x(4r+1) W_N^{(4r+1)(k+3N/4)} \\ &+ \sum_{r=0}^{N/4-1} x(4r+2) W_N^{(4r+2)(k+3N/4)} \\ &+ \sum_{r=0}^{N/4-1} x(4r+3) W_N^{(4r+3)(k+3N/4)} \end{aligned}$$

# 关于 FFT 的讨论

为何基-2, 4 可以省运算量?

$$X(k + \frac{lN}{P}) = \sum_{m=0}^{P-1} \sum_{n=0}^{\frac{N}{P}-1} x(nP + m) W_N^{(nP+m)(k+\frac{lN}{P})}, 0 \leq l \leq P-1$$

- $P \neq 2, 4$  时,  $W_N^{\frac{lmN}{P}}$  必须通过计算获得!  $m \neq 0$  列不能获得运算量降低
- $P = 2, 4$  时,  $W_N^{\frac{lmN}{P}}$  为  $\pm 1, \pm j$  无需计算乘法,  $m \neq 0$  的列依然可以降低运算量

# 傅里叶变换应用

实序列DFT

线性调频  $Z$ 变换

DFT实现线性滤波器

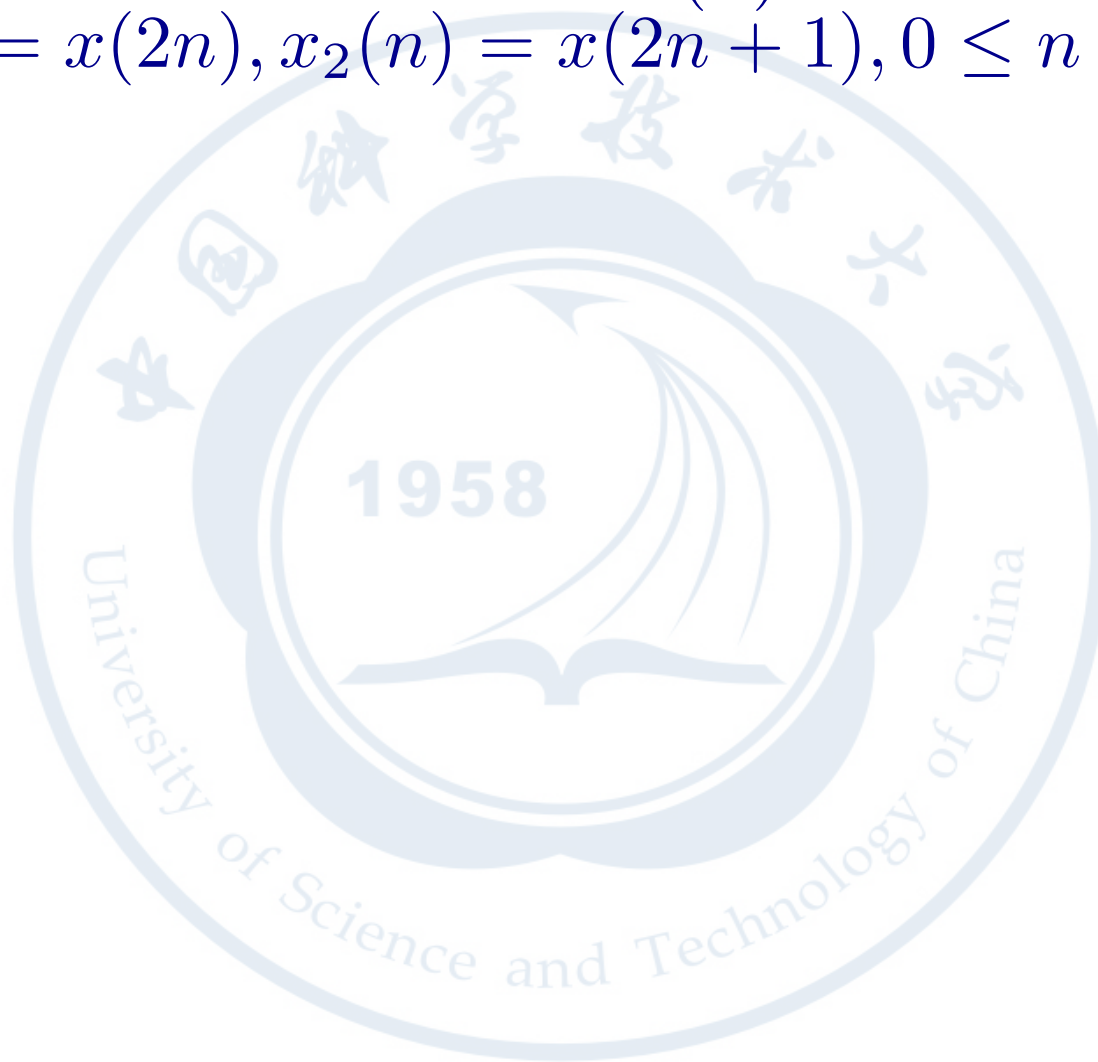




# 实序列 FFT的计算

考虑一个长度为  $2^{m+1}$  的实序列  $x(n)$  的 FFT:

$$x_1(n) = x(2n), x_2(n) = x(2n+1), 0 \leq n \leq 2^m - 1$$



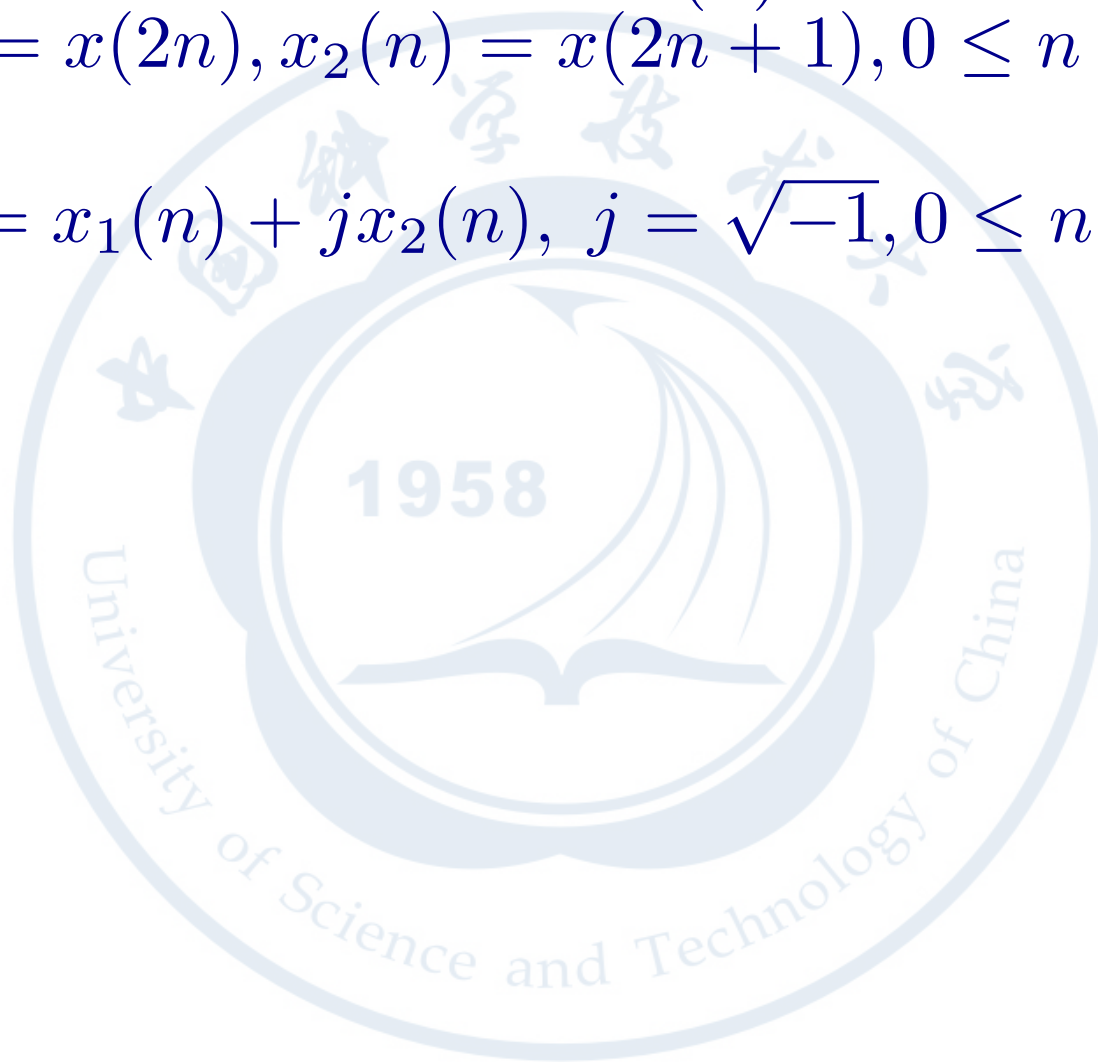
# 实序列 FFT的计算

考虑一个长度为  $2^{m+1}$  的实序列  $x(n)$  的 FFT:

$$x_1(n) = x(2n), x_2(n) = x(2n+1), 0 \leq n \leq 2^m - 1$$

定义:

$$x_0(n) = x_1(n) + jx_2(n), j = \sqrt{-1}, 0 \leq n \leq 2^m - 1$$



# 实序列 FFT的计算

考虑一个长度为  $2^{m+1}$  的实序列  $x(n)$  的 FFT:

$$x_1(n) = x(2n), x_2(n) = x(2n+1), 0 \leq n \leq 2^m - 1$$

定义:

$$x_0(n) = x_1(n) + jx_2(n), j = \sqrt{-1}, 0 \leq n \leq 2^m - 1$$

所以:

$$x_1(n) = x_0(n) + x_1^*(n), x_2(n) = \frac{1}{2j} (x_0(n) - x_1^*(n))$$

# 实序列 FFT的计算

考虑一个长度为  $2^{m+1}$  的实序列  $x(n)$  的 FFT:

$$x_1(n) = x(2n), x_2(n) = x(2n+1), 0 \leq n \leq 2^m - 1$$

定义:

$$x_0(n) = x_1(n) + jx_2(n), j = \sqrt{-1}, 0 \leq n \leq 2^m - 1$$

所以:

$$x_1(n) = x_0(n) + x_1^*(n), x_2(n) = \frac{1}{2j} (x_0(n) - x_1^*(n))$$

进一步:

$$X_1(k) = \text{DFT}[x_1(n)] = X_0(k) + X_0^*(2^m - k)$$

$$X_2(k) = \text{DFT}[x_2(n)] = X_0(k) + X_0^*(2^m - k)$$

其中  $X_0(k) = \text{DFT}[x_0(n)]$

# 实序列 FFT的计算

考虑一个长度为  $2^{m+1}$  的实序列  $x(n)$  的 FFT:

$$x_1(n) = x(2n), x_2(n) = x(2n+1), 0 \leq n \leq 2^m - 1$$

定义:

$$x_0(n) = x_1(n) + jx_2(n), j = \sqrt{-1}, 0 \leq n \leq 2^m - 1$$

所以:

$$x_1(n) = x_0(n) + x_1^*(n), x_2(n) = \frac{1}{2j} (x_0(n) - x_1^*(n))$$

进一步:

$$X_1(k) = \text{DFT}[x_1(n)] = X_0(k) + X_0^*(2^m - k)$$

$$X_2(k) = \text{DFT}[x_2(n)] = X_0(k) - X_0^*(2^m - k)$$

其中  $X_0(k) = \text{DFT}[x_0(n)]$

于是:

$$\begin{aligned} X(k) &= X_1(k) + W_{2^{m+1}}^k X_2(k) \\ X(k + N) &= X_1(k) - W_{2^{m+1}}^k X_2(k) \end{aligned}$$

# FFT实现线性调频 $z$ 变换

## Motivation:

- 原有的基于 **DIT,DIF**的快速付立叶变换均需要  $N$ 是复合数，研究目标也是快速的对  $z$ 变换单位圆  $|z| = 1$ 上的结果进行**等间隔取样**





# FFT实现线性调频 $z$ 变换

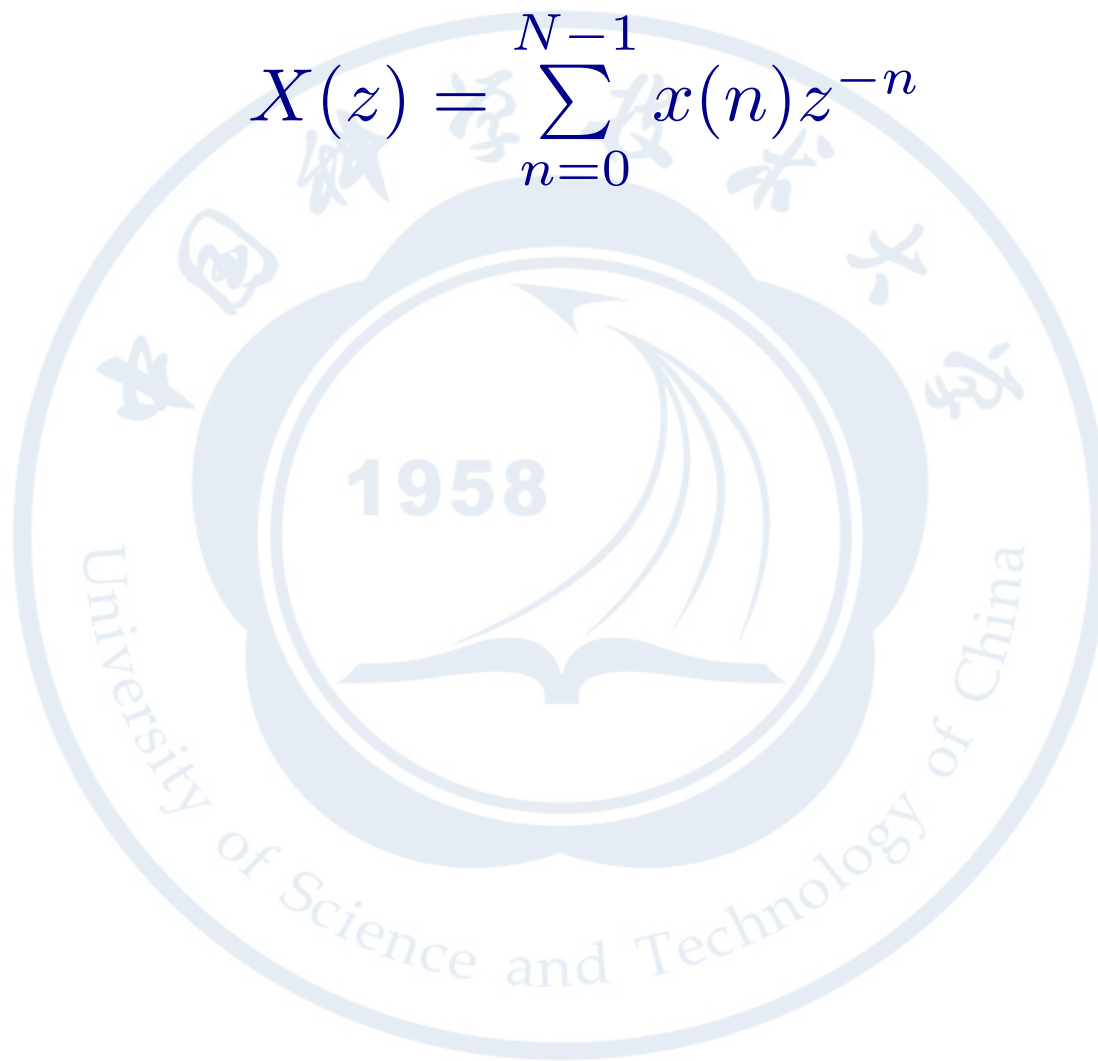
## Motivation:

- 原有的基于 **DIT,DIF**的快速付立叶变换均需要  $N$ 是复合数，研究目标也是快速的对  $z$ 变换单位圆  $|z| = 1$ 上的结果进行**等间隔取样**
- 对于  $z$ 变换取样点不在单元圆上等间隔取样，或者  $N$ 是素数是如何实现对  $|z| = 1$ 的快速等间隔取样计算问题

# 线性调频 Z 变换 (Chirp Z Transform) 系统模型

对于一个序列  $x(n), 0 \leq n \leq N-1$ , 其 Z 变换可以表示为:

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



# 线性调频 Z 变换 (Chirp Z Transform) 系统模型

对于一个序列  $x(n), 0 \leq n \leq N-1$ , 其  $Z$  变换可以表示为:

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

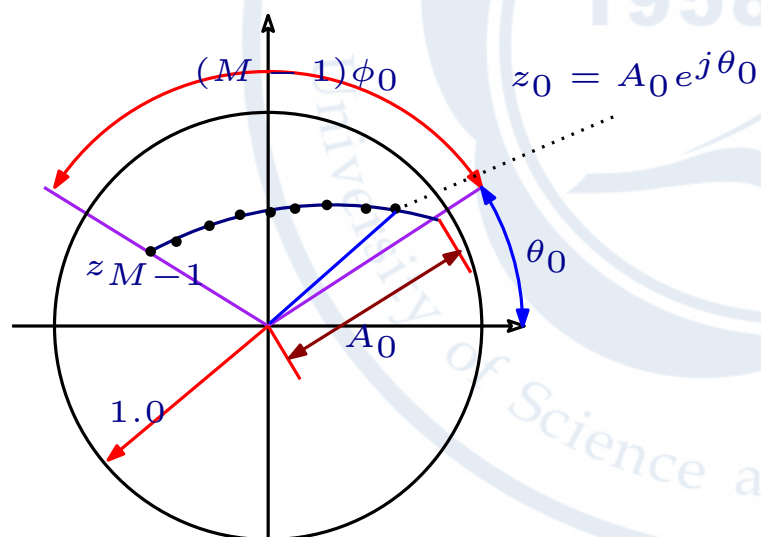
问题: 如何对  $z_k = AW^{-k}, 0 \leq k \leq M-1$  的序列点进行的取样结果快速计算。这里  $M \neq N$ ,  $A = A_0 e^{j\theta_0}, W = W_0 e^{-j\phi_0}$

# 线性调频 Z 变换 (Chirp Z Transform) 系统模型

对于一个序列  $x(n), 0 \leq n \leq N-1$ , 其  $Z$  变换可以表示为:

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

问题: 如何对  $z_k = AW^{-k}, 0 \leq k \leq M-1$  的序列点进行的取样结果快速计算。这里  $M \neq N$ ,  $A = A_0 e^{j\theta_0}, W = W_0 e^{-j\phi_0}$



$$z_k = A_0 W_0^{-k} e^{j(\theta_0 + k\phi_0)}$$

$$M = N$$

$$A = A_0 e^{j\theta_0}$$

$$W = W_0 e^{j\phi_0} = e^{-j\frac{2\pi}{N}}$$

CZT  $\longrightarrow$  DFT

线性调频  $z$  变换的  $z$  平面曲线

# 利用 FFT 计算线性调频 Z 变换

线性调频 Z 变换:

$$X(z_k) = \sum_{n=0}^{N-1} x(n) z^{-n} \Big|_{z_k = AW^{-k}}, \quad 0 \leq k \leq M-1$$

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

# 利用 FFT 计算线性调频 Z 变换

线性调频 Z 变换:

$$X(z_k) = \sum_{n=0}^{N-1} x(n) z^{-n} \Big|_{z_k = AW^{-k}}, \quad 0 \leq k \leq M-1$$

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

注意到,  $nk = \frac{1}{2}(n^2 + k^2 - (n-k)^2)$



# 利用 FFT 计算线性调频 Z 变换

线性调频 Z 变换:

$$X(z_k) = \sum_{n=0}^{N-1} x(n) z^{-n} \Big|_{z_k = AW^{-k}}, \quad 0 \leq k \leq M-1$$

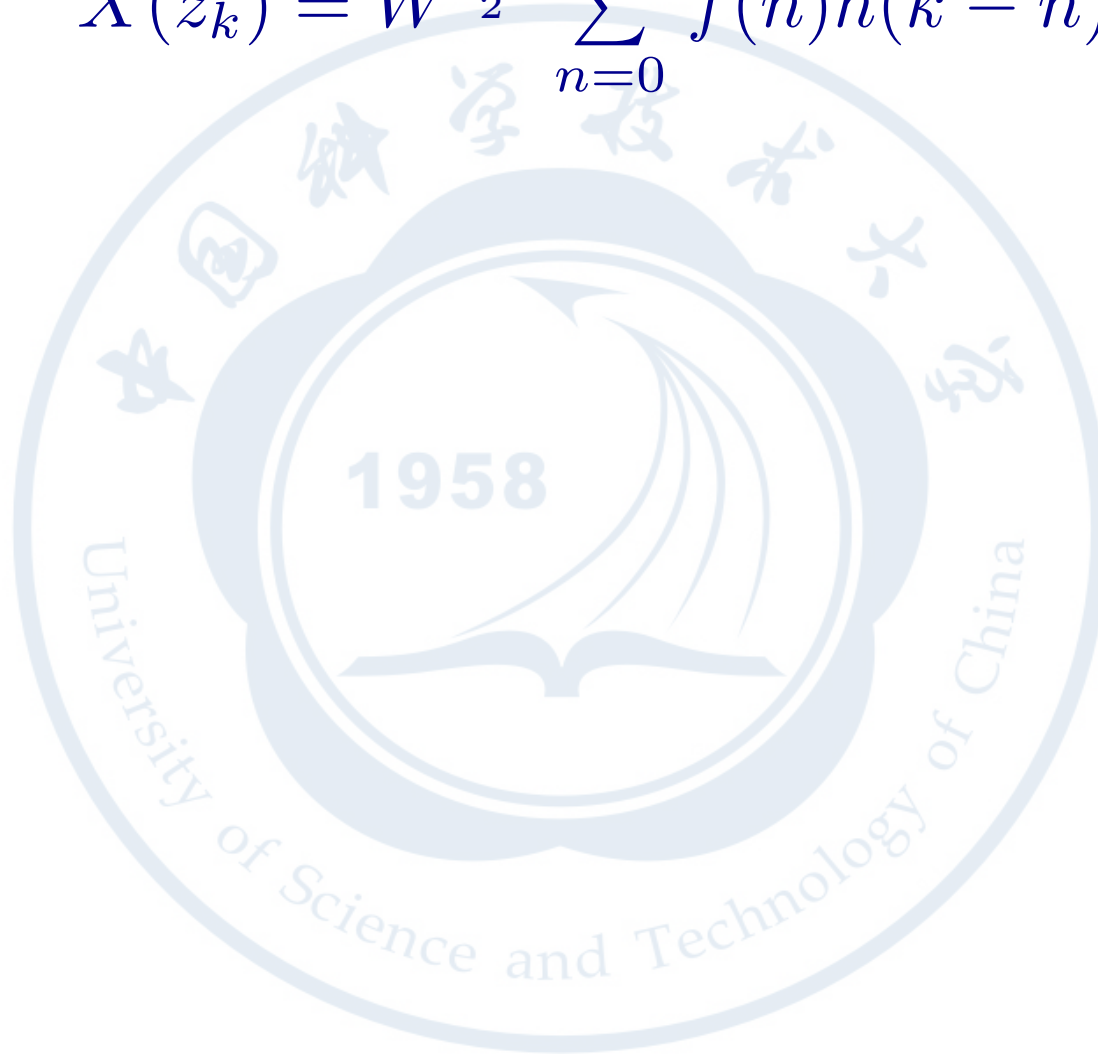
$$X(z_k) = \sum_{n=0}^{N-1} x(n) A^{-n} W^{nk}, \quad 0 \leq k \leq M-1$$

注意到,  $nk = \frac{1}{2}(n^2 + k^2 - (n-k)^2)$

$$\begin{aligned} X(z_k) &= \sum_{n=0}^{N-1} x(n) A^{-n} W^{\frac{n^2}{2}} W^{-\frac{k^2}{2}} W^{-\frac{(k-n)^2}{2}} \\ &= W^{\frac{k^2}{2}} \cdot \underbrace{\sum_{n=0}^{N-1} \left[ x(n) A^{-n} W^{\frac{n^2}{2}} \right]}_{f(n)} \underbrace{W^{-\frac{(k-n)^2}{2}}}_{h(n)=W^{-\frac{n^2}{2}}} \end{aligned}$$

# 基于 FFT 的线性调频 z 变换快速算法

$$X(z_k) = W^{\frac{k^2}{2}} \sum_{n=0}^{N-1} f(n)h(k-n)$$



# 基于 FFT 的线性调频 z 变换快速算法

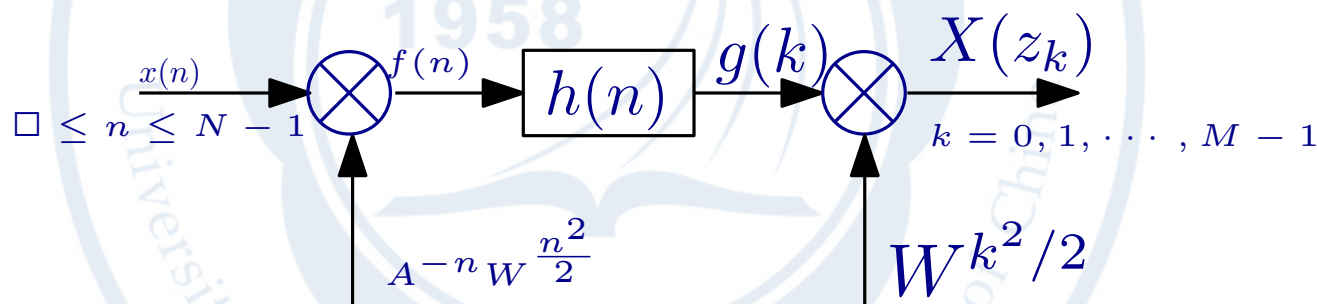
$$X(z_k) = W^{\frac{k^2}{2}} \sum_{n=0}^{N-1} f(n)h(k-n)$$

$z_k$  对应  $z$  变换数值  $X(z_k)$  可以表征为  $f(n)$  和  $h(n)$  的离散卷积与  $W^{k^2/2}$  的乘积，离散卷积可以基于圆周卷积利用 FFT 计算

# 基于 FFT 的线性调频 z 变换快速算法

$$X(z_k) = W^{\frac{k^2}{2}} \sum_{n=0}^{N-1} f(n)h(k-n)$$

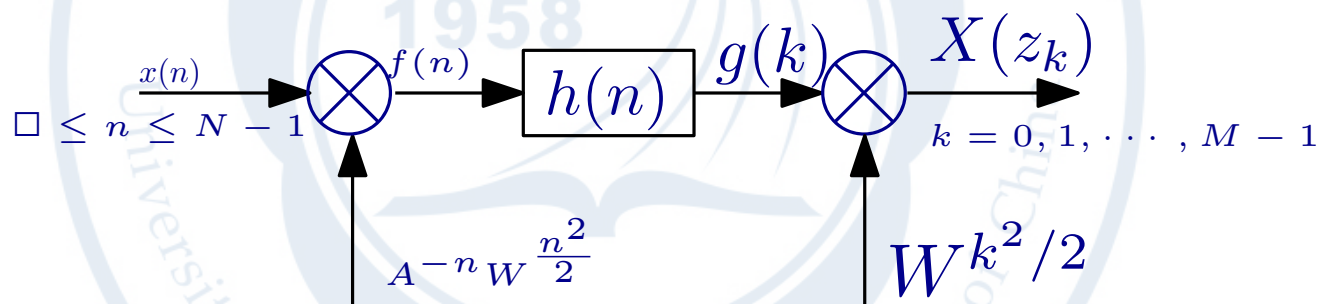
$z_k$  对应  $z$  变换数值  $X(z_k)$  可以表征为  $f(n)$  和  $h(n)$  的离散卷积与  $W^{k^2/2}$  的乘积，离散卷积可以基于圆周卷积利用 FFT 计算



# 基于 FFT 的线性调频 z 变换快速算法

$$X(z_k) = W^{\frac{k^2}{2}} \sum_{n=0}^{N-1} f(n)h(k-n)$$

$z_k$  对应  $z$  变换数值  $X(z_k)$  可以表征为  $f(n)$  和  $h(n)$  的离散卷积与  $W^{k^2/2}$  的乘积，离散卷积可以基于圆周卷积利用 FFT 计算

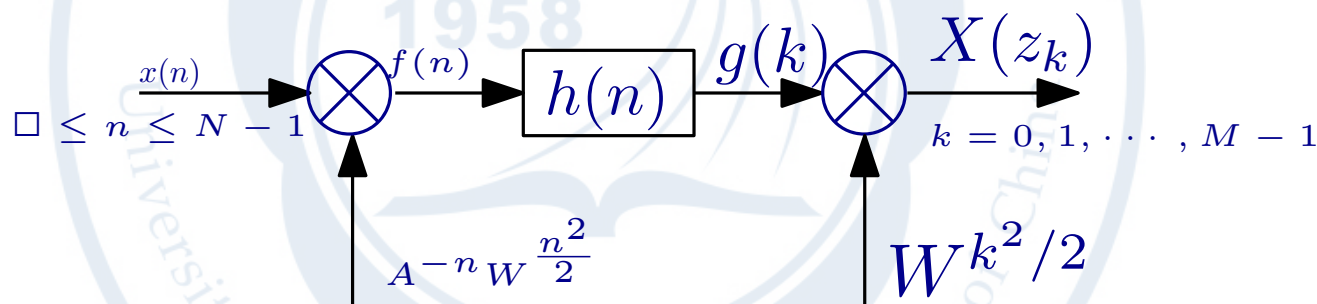


**问题：**  $X_k$  中  $0 \leq k \leq M-1$ ,  $x(n)$  中  $n$  的范围是  $0 \leq n \leq N-1$ ,  $h(n)$  的取值范围？

# 基于 FFT 的线性调频 z 变换快速算法

$$X(z_k) = W^{\frac{k^2}{2}} \sum_{n=0}^{N-1} f(n)h(k-n)$$

$z_k$  对应  $z$  变换数值  $X(z_k)$  可以表征为  $f(n)$  和  $h(n)$  的离散卷积与  $W^{k^2/2}$  的乘积，离散卷积可以基于圆周卷积利用 FFT 计算



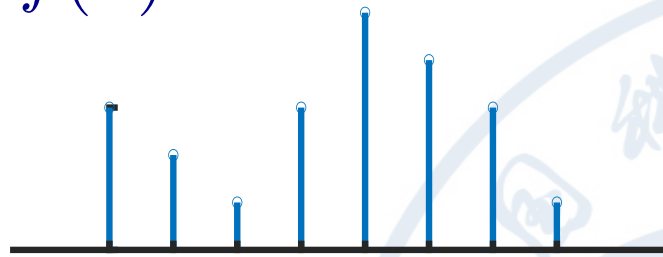
**问题:**  $X_k$  中  $0 \leq k \leq M-1$ ,  $x(n)$  中  $n$  的范围是  $0 \leq n \leq N-1$ ,  $h(n)$  的取值范围?

$$0 \leq k \leq M-1, 0 \leq n \leq N-1 \rightarrow -(N-1) \leq k-n \leq M-1 \\ \rightarrow h(n), -(N-1) \leq n \leq M-1$$



# 基于 FFT 的线性调频 Z 变换

**Step 1:** 根据给定的  $A_0, \theta_0, W_0, \phi_0$ , 求取  $A^{-n} W^{\frac{n^2}{2}}$ , 进一步得到  $f(n)$



$x(n)$



$$f(n) = \begin{cases} A^{-n} W^{\frac{n^2}{2}} x(n) & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases}$$

提示:

$f(n)$  长度  $N$ ,  $h(n)$  长度  $N + M_1$ , 则线性卷积长度  $2N + M - 2$

# 利用 FFT 计算线性调频 Z 变换

**Step 2:** 选择最小的  $L$ , 使其满足  $L > (M + N - 1)$ , 而且  $L = 2^m$ , 以使用基-2FFT 算法求得  $f(n)$  与  $h(n)$  卷积



# 利用 FFT 计算线性调频 Z 变换

**Step 2:** 选择最小的  $L$ , 使其满足  $L > (M + N - 1)$ , 而且  $L = 2^m$ , 以使用基-2FFT 算法求得  $f(n)$  与  $h(n)$  卷积

**Problem:**

付立叶变换选择的是  $L > N + M - 1$  而不是  $2N + M - 1$ , why?

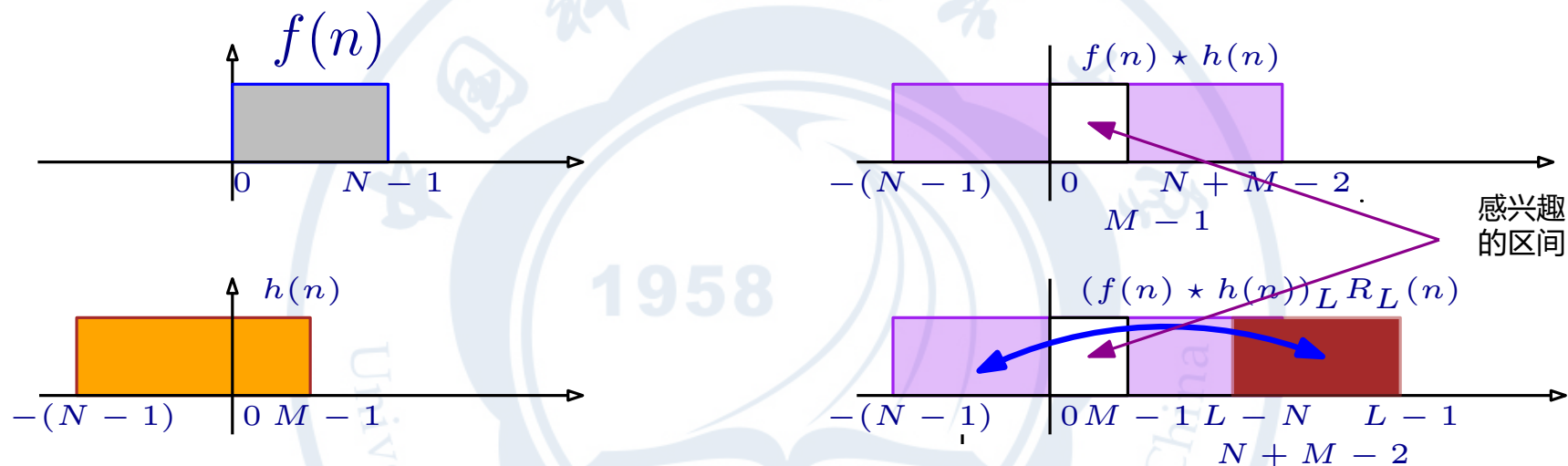
Note:

输入序列	起点	终点	长度
$x(n)$	0	$N - 1$	$N$
$f(n)$	0	$N - 1$	$N$
$h(n)$	$-(N - 1)$	$M - 1$	$N + M - 1$
$f(n) * h(n)$	$-(N - 1)$	$N + M - 1$	$2N + M - 1$

# 利用 FFT 计算线性调频 Z 变换

## Note 2:

我们感兴趣的区域并不是  $f(n) \star h(n)$  的整个区域  $-(N-1) \leq n \leq 2N+M-1$ , 仅仅关注  $0 \leq n \leq M-1$ 。



结论:  $L-N > M-1$  时, 目标区域 (白色) 不与其他其他区域发生混叠, 此时对应  $L > N+M-1$

# 基于 FFT 的线性调频 Z 变换算法

**Step 3:** 采用补零的方法将 Step 1 的的  $f(n)$  变为列长为  $L$  的序列, 求得其付立叶变换结果  $F(r)$

**Step 4:** 将  $h(n)$  以  $L$  周期进行延拓, 并取主值区间:

$$h(n) = \begin{cases} W^{\frac{n^2}{2}} & 0 \leq n \leq M-1 \\ W^{\frac{(L-n)^2}{2}} & L-N+1 \leq n \leq L-1 \\ - & M \leq n \leq L-N \end{cases}$$

根据  $h(n)$  求取  $L$  点 DFT 结果  $H(r)$

**Step 5:** 令  $G(r) = F(r)H(r)$ ,  $g(k) = \text{IDFT}(r)$

**Step 6:**  $X(z_k) = W^{\frac{k^2}{2}} g(k)$ ,  $0 \leq k \leq N-1$

# 基于 FFT 的线性调频 Z 变换算法

## 运算复杂度分析

步骤	完成工作	运算开销
1	$x(n) \rightarrow f(n)$	$N$
2	$f(n) \rightarrow F(r)$	$1/2L \log_2 L$
3	$h(n) \rightarrow H(r)$	—
4	$G(r) = H(r)F(r)$	$L$
5	$G(r) \rightarrow g(k)$	$1/2L \log_2 L$
6	$X(z_k) = g(k)W^{k^2/2}$	$M - 1$
总计		$L \log_2 L + L + M$

运算复杂度对比举例（思考不同的  $N$  和  $M$  的影响）

$N$	$M$	原始算法	基于 FFT
50	30	1500	976
50	20	1000	950



# 利用 FFT 计算线性卷积

## 回顾:

线性卷积与线性卷积关系满足下述关系, 若序列  $x_1(n), 0 \leq n \leq N_1 - 1$  和  $x_2(n), 0 \leq n \leq N_2 - 1$  的圆周卷积  $x_1(n) \otimes x_2(n)$  的周期  $L$  满足  $L > N_1 + N_2 - 1$ , 则有:

$$x_1(n) \star x_2(n) = (x_1(n) \otimes x_2(n))_L R_N(n)$$

## 思考:

对于一个信号其输入  $x(n)$  可能是无限长  $-\infty < n < \infty$  与单位冲击相应  $h(n)$  卷积该如何计算以降低复杂度。

## 问题:

一个序列无限长, 他参与的圆周卷积的要想和现行卷积的长度一致, 意味着圆周卷积长度也要无限长, 工程上无法实现, 也意味着输出结果的时延也无穷大。

# 重叠相加法

$$x(n) = \sum_{i=-\infty}^{\infty} x_i(n)$$

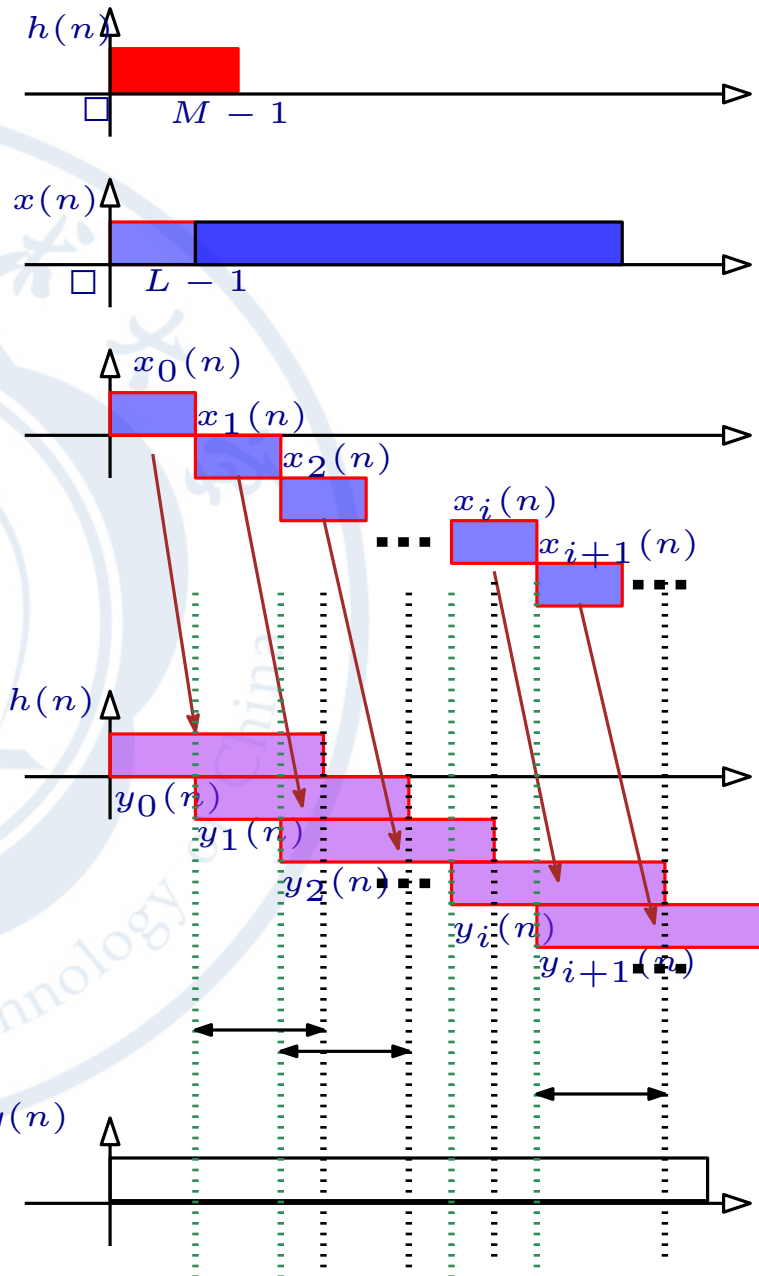
$$x_i(n) = \begin{cases} x(n) & (i-1)L \leq n < iL - 1 \\ 0 & \text{otherwise} \end{cases}$$

卷积的线性定理:

$$y(n) = x(n) \star h(n)$$

$$\square \sum_{i=-\infty}^{\infty} x_i(n) \star h(n)$$

$$\square \sum_{i=-\infty}^{\infty} y_i(n)$$



重叠相加法

分段卷积, 重叠部分相加

# 重叠保留法

## 预先准备:

对于一些长度为  $L$  的序列  $x(n)$ , 与一个长度为  $M$  的序列  $h(n)$  进行线性卷积结果与长度为  $N$  的圆周卷积关系为:

$$x(n) \otimes h(n) = (x(n) \star h(n))_N R_N(n)$$

特别地, 当  $N = L (L > M)$  时, 上述操作引起混叠, 但是仍然有  $L - M + 1$  点结果与线性卷积结果一致。

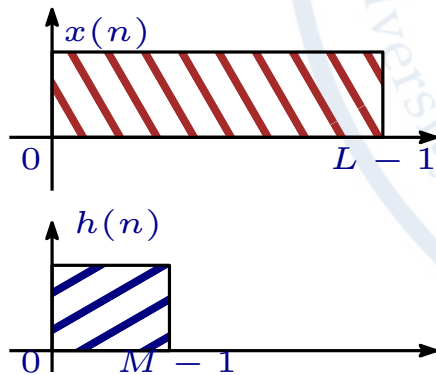
# 重叠保留法

## 预先准备:

对于一些长度为  $L$  的序列  $x(n)$ , 与一个长度为  $M$  的序列  $h(n)$  进行线性卷积结果与长度为  $N$  的圆周卷积关系为:

$$x(n) \otimes h(n) = (x(n) \star h(n))_N R_N(n)$$

特别地, 当  $N = L (L > M)$  时, 上述操作引起混叠, 但是仍然有  $L - M + 1$  点结果与线性卷积结果一致。



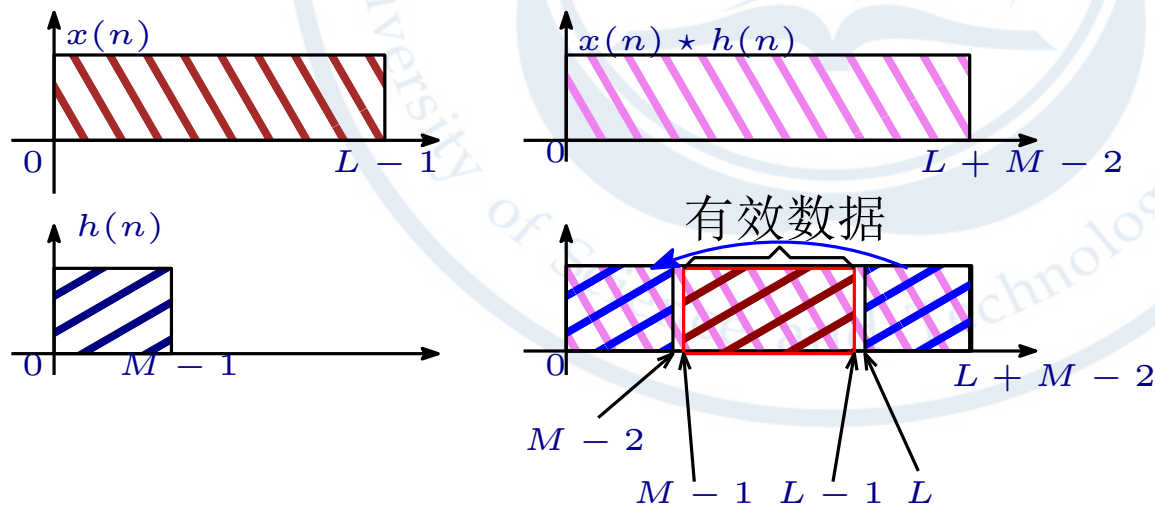
# 重叠保留法

## 预先准备:

对于一些长度为  $L$  的序列  $x(n)$ , 与一个长度为  $M$  的序列  $h(n)$  进行线性卷积结果与长度为  $N$  的圆周卷积关系为:

$$x(n) \otimes h(n) = (x(n) \star h(n))_N R_N(n)$$

特别地, 当  $N = L(L > M)$  时, 上述操作引起混叠, 但是仍然有  $L - M + 1$  点结果与线性卷积结果一致。



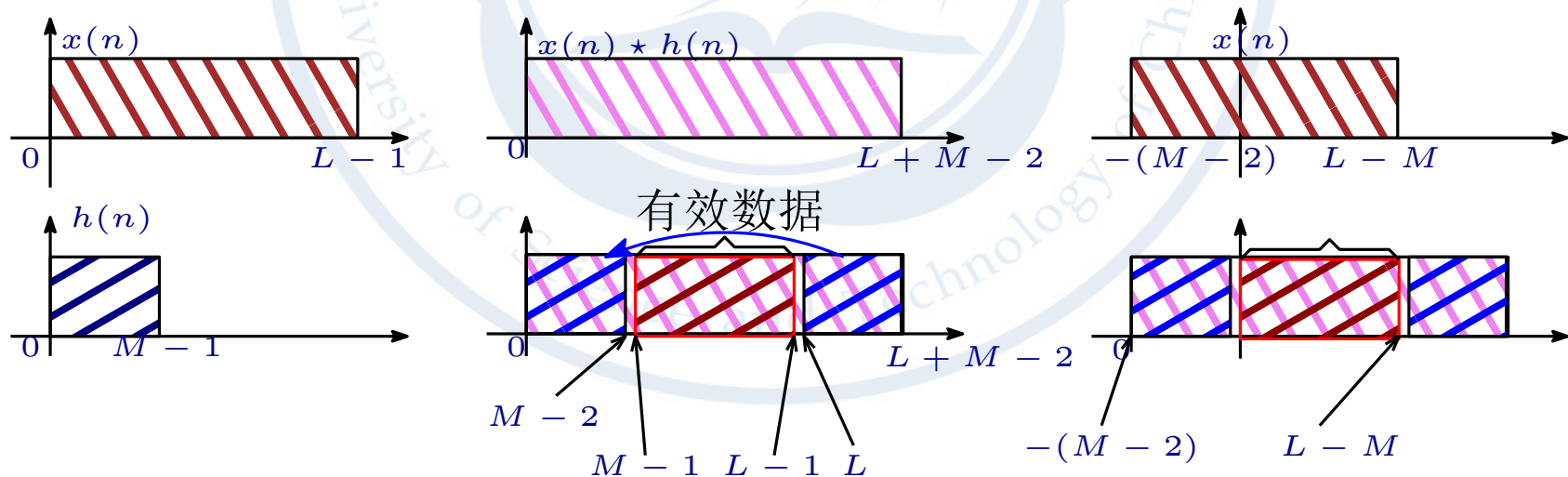
# 重叠保留法

## 预先准备:

对于一些长度为  $L$  的序列  $x(n)$ , 与一个长度为  $M$  的序列  $h(n)$  进行线性卷积结果与长度为  $N$  的圆周卷积关系为:

$$x(n) \otimes h(n) = (x(n) \star h(n))_N R_N(n)$$

特别地, 当  $N = L (L > M)$  时, 上述操作引起混叠, 但是仍然有  $L - M + 1$  点结果与线性卷积结果一致。

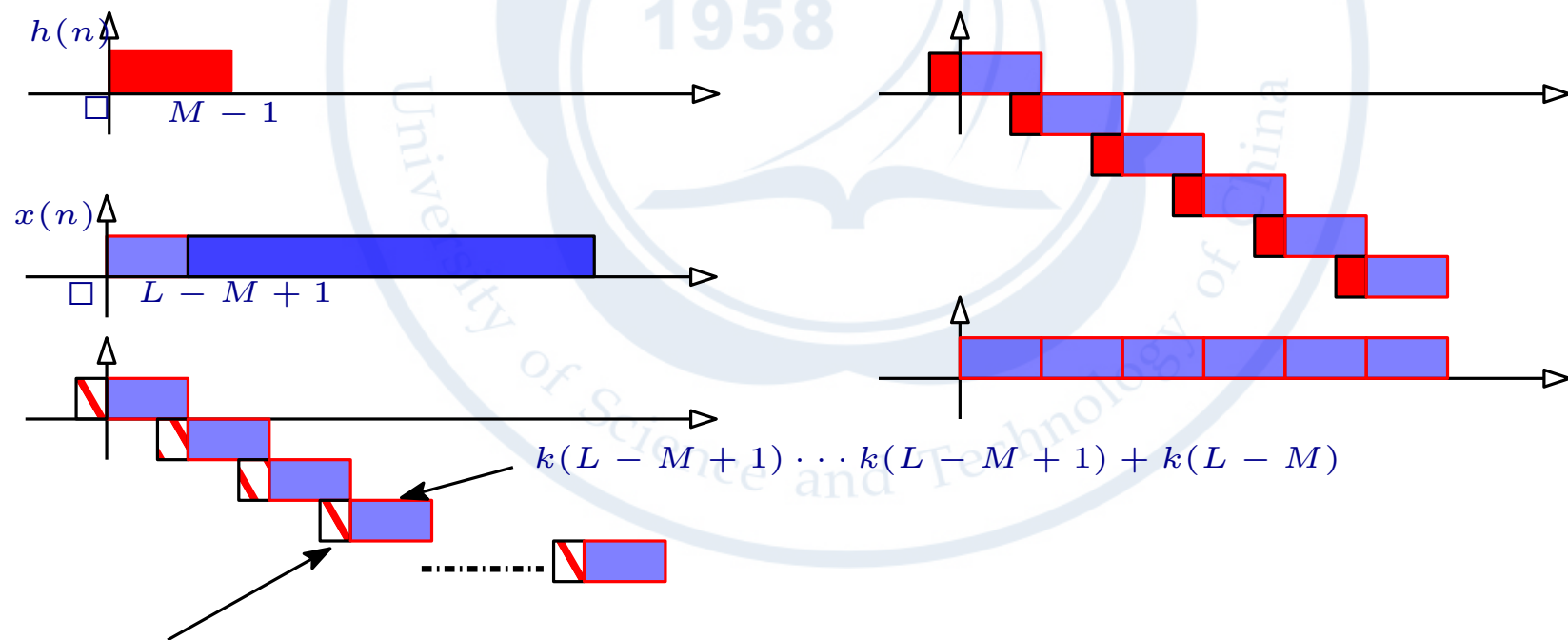




# 重叠保留法

## 思路:

求取  $x(n)$  与长度为  $M$  的序列  $h(n)$  的线性卷积, 将输入序列  $x(n)$  分为若干段, 每段长度长度为  $L - M + 1$ , 并保留前面的本段序列之前的  $M - 1$  个序列值与  $h(n)$  做  $L$  点圆周卷积, 取结果的第  $M - 1$  点到  $L - 1$  并拼接



$$k(L - M + 1) - (M - 2) \dots k(L - M + 1) - 1$$

# 重叠保留法和重叠相加法运算量对比

- 重叠保留法和重叠相加法都是每  $L$  点 DFT 实际得到  $L - M + 1$  个线性卷积点
- 重叠相加法每次  $L$  点操作实际上需要额外的加法操作，其操作次数是超过重叠保留法的，但是差别很小



# 线性相关的频域计算方法

问题：列长为  $N_1$  的实序列  $x_1(n)$

$$y(n) = \sum_{k=0}^{N_1-1} x_1(k)x_2(n+k)$$

相关结果长度  $N_1 + N_2 - 1$ ，为避免周期项引入的混叠，对  $x_1(n)$  和  $x_2(n)$  需要补零， $N = 2^v \geq N_1 + N_2 - 1$

$$\tilde{x}_1(n) = \begin{cases} x_1(n) & n = 0, 1, \dots, N_1 - 1 \\ 0 & n = N - 1, \dots, N - 1 \end{cases}$$

$$\tilde{x}_2(n) = \begin{cases} x_2(n) & n = 0, 1, \dots, N_2 - 1 \\ 0 & n = N - 1, \dots, N - 1 \end{cases}$$

$$\tilde{x}_1(n) \longrightarrow X_1(k) \quad Y(k) = X_1^*(k)X_2(k) \longrightarrow y(n)$$

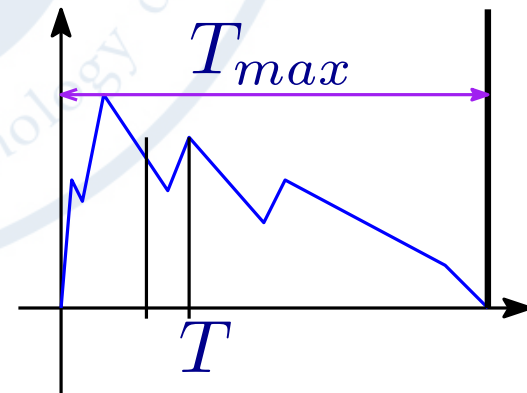
$$\tilde{x}_2(n) \longrightarrow X_2(k)$$

# FFT在 OFDM系统中的应用

OFDM调制技术：宽带、高速无线通信  
(Orthogonal Frequency Division Multiplexing)

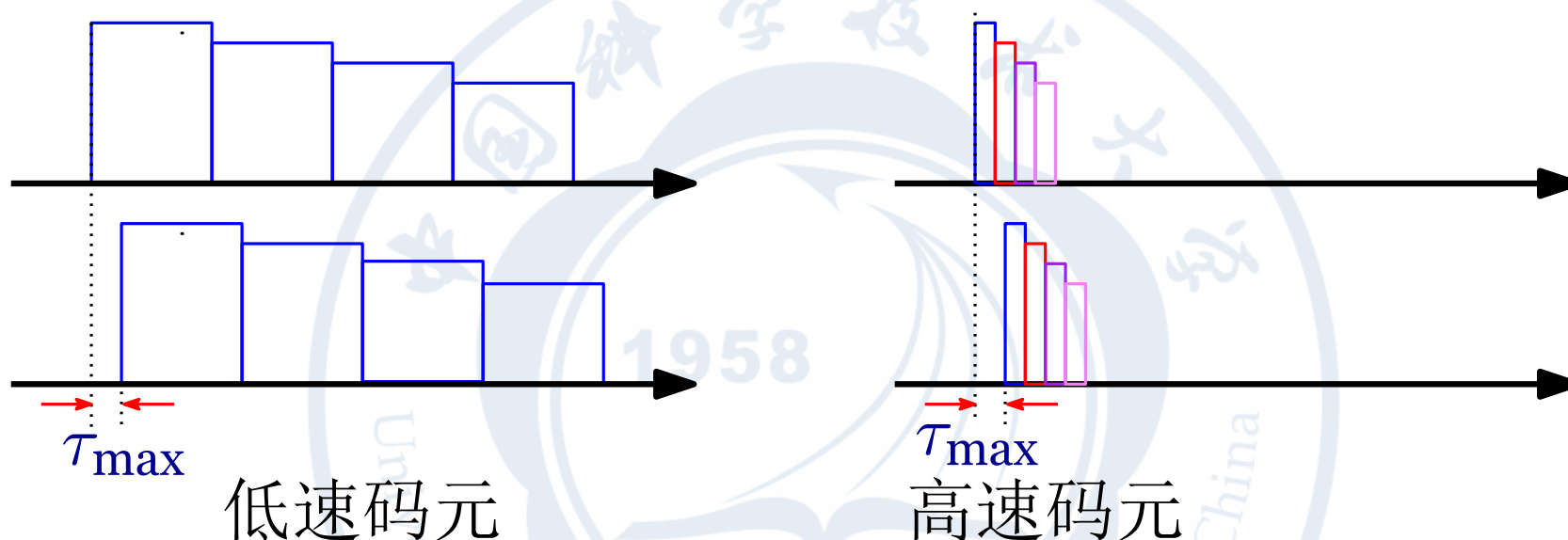
单载波调制：

- 时延扩展，产生符号间干扰
- 带宽过宽，频率产生性衰落
- 信道均衡困难



# FFT在 OFDM系统中的应用

相同多径时延下不同传输速率的符号间干扰对比

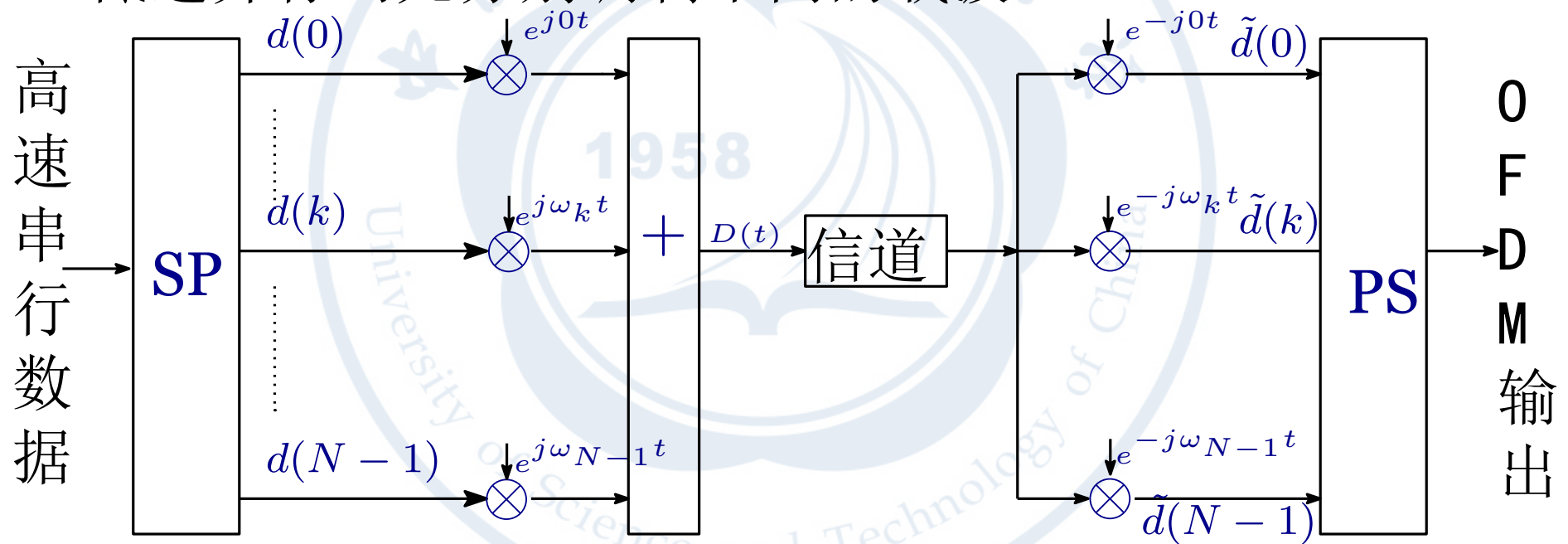


大带宽信号对应的高速传输信号会收到不同  
时延的多径分量干扰，传输性能下降

# FFT在 OFDM系统中的应用

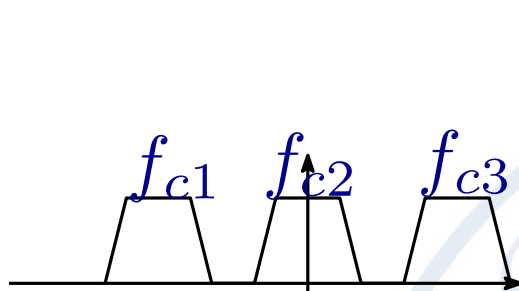
## 多载波调制

- 高速串行码元转化为低速并行码元
- 低速并行码元分别调制不同的载波



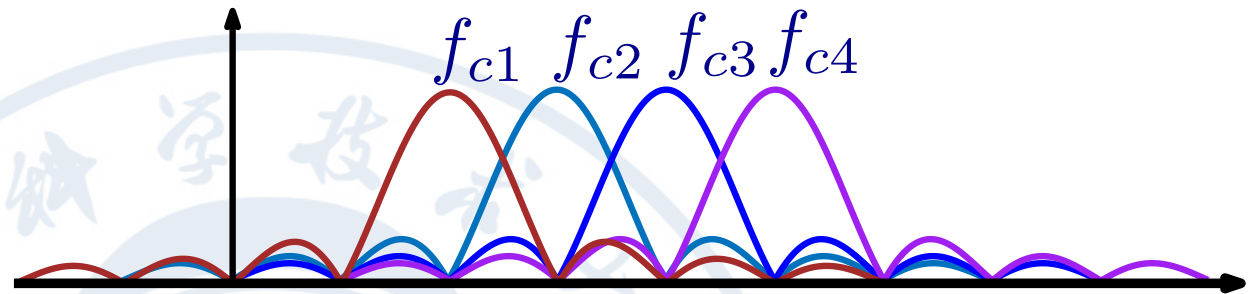


# FFT在 OFDM系统中的应用



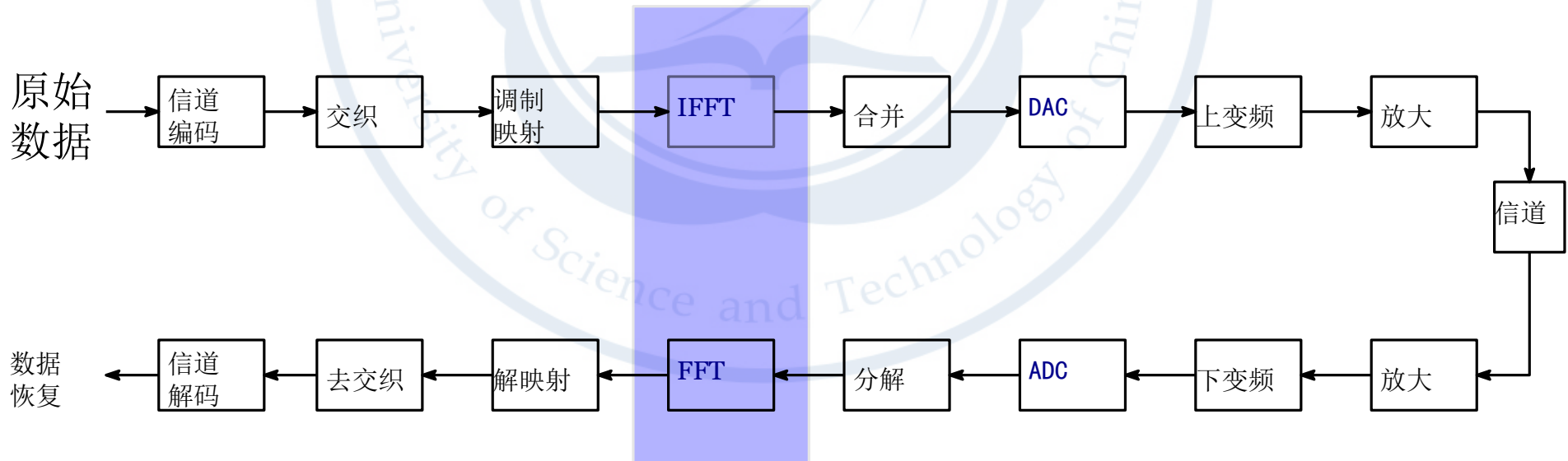
多载波通信系统

- 频谱效率低
- 实现困难



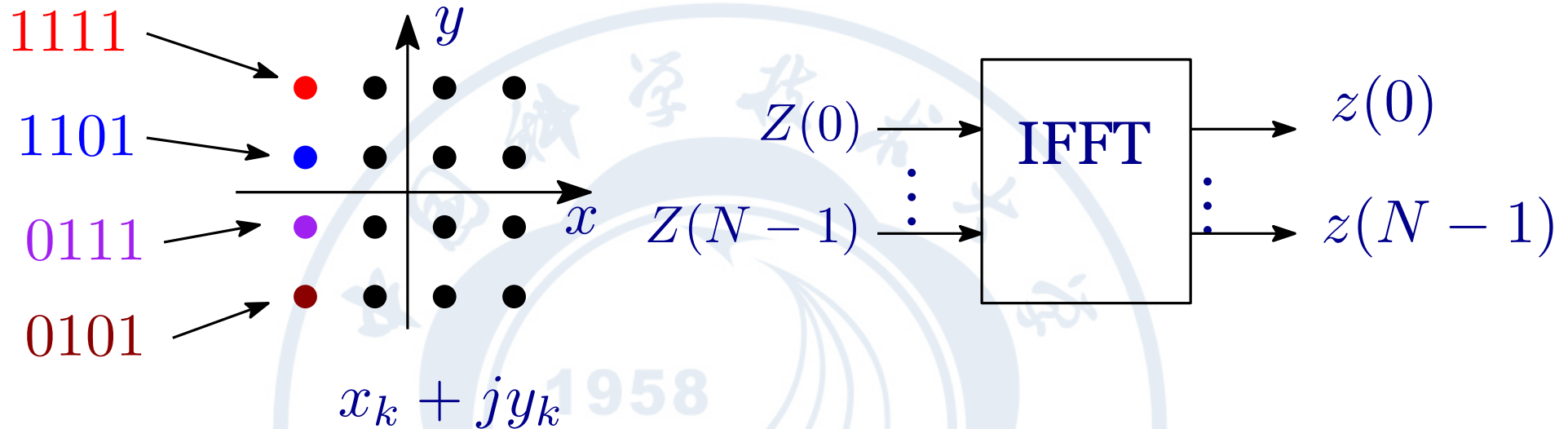
OFDM系统

- 频率效率高
- 可用 FFT实现



# FFT在 OFDM系统中的应用

## Mapping Method: m-QAM



$M \log_2 M$  个比特映射为  $M$  个星座点  $Z_k = x_k + jy_k$

- 数据速率  $R(bps)$ , 系统带宽  $W(Hz)$ ,  $M = 2^{R/W}$
- 为了克服信道多径时延, 在信元之间加保护间隔

# 基于 FFT 的捕获同步技术

## CDMA 接收机原理

