



3 快速傅立叶变换 (FFT)



3.1 问题的提出

3.1 问题的提出

已知 N 点有限长序列 $x(n)$ 的DFT为

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

直接计算 N 点谱线,

$$\begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} W_N^0 & W_N^0 & \dots & W_N^0 \\ W_N^0 & W_N^1 & \dots & W_N^{(N-1)} \\ \vdots & \vdots & & \vdots \\ W_N^0 & W_N^{(N-1)} & \dots & W_N^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}$$

3.1 问题的提出

原则上，直接计算需要 N^2 次复数乘法。

复数乘法：

$$(a_1 + jb_1)(a_2 + jb_2) = a_1a_2 - b_1b_2 + j(a_1b_2 + a_2b_1)$$

N 点DFT： $4 N^2$ 次实数乘法。

3.1 问题的提出

- 减少运算量的两个途径

- 1) 将较长序列的DFT, 分解为若干较短序列的DFT

原因：乘法运算量与序列长度的平方成正比

3.1 问题的提出

2)利用指数因子的特性（周期性，对称性）

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^1 & W_4^2 & W_4^3 \\ W_4^0 & W_4^2 & W_4^4 & W_4^6 \\ W_4^0 & W_4^3 & W_4^6 & W_4^9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$

3.1 问题的提出

$$W_4^0 = 1 \quad W_4^1 = -j \quad W_4^2 = -1 \quad W_4^4 = 1$$

$$W_4^3 = W_4^2 \cdot W_4^1 = j \quad W_4^6 = W_4^2 \cdot W_4^4 = -1$$

$$W_4^9 = W_4^8 \cdot W_4^1 = -j$$

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

结论：完成4点DFT可以不用乘法

3.2 基2的DIT-FFT(按时间抽取的FFT)

Decimation In Time

3.2 基2的DIT-FFT(按时间抽取的FFT)

(1) N 为偶数

将 $x(n)$ 分为奇偶两组,

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

3.2 基2的DIT-FFT(按时间抽取的FFT)

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

$$\begin{aligned} \therefore X(k) &= \sum_{r=0}^{N/2-1} x(2r)W_{N/2}^{rk} + W_N^k \sum_{r=0}^{N/2-1} x(2r+1)W_{N/2}^{rk} \\ &= G(k) + W_N^k H(k) \quad k=0, 1, 2, \dots, N-1 \end{aligned}$$

3.2 基2的DIT-FFT(按时间抽取的FFT)

这里,

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

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

$G(k)$ 与 $H(k)$ 均为 $N/2$ 点 DFT, 以 $N/2$ 为周期

3.2 基2的DIT-FFT(按时间抽取的FFT)

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

$$H(k + \frac{N}{2}) = H(k) \quad k = 0, 1, \dots, \frac{N}{2} - 1$$

另外, $W_N^{k+N/2} = -W_N^k$

3.2 基2的DIT-FFT(按时间抽取的FFT)

$$\left. \begin{aligned} \therefore X(k) &= G(k) + W_N^k H(k) \\ X\left(k + \frac{N}{2}\right) &= G(k) - W_N^k H(k) \end{aligned} \right\} k = 0, 1, 2, \dots, \frac{N}{2} - 1$$

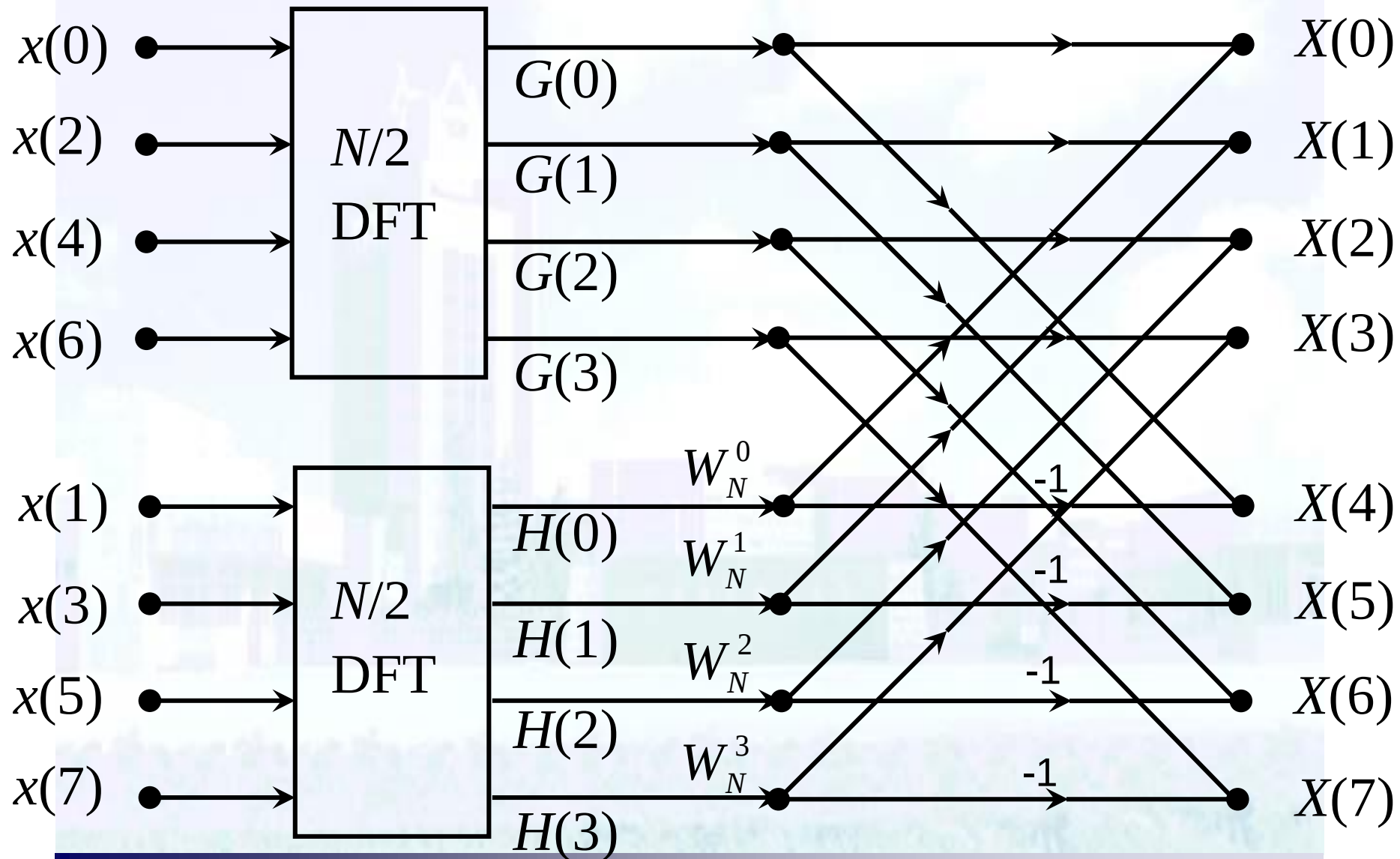
运算量估计:

2个的 $N/2$ 长DFT, 系数乘法次数 $N/2$,

总的乘法运算量: $\frac{1}{2} N^2 + \frac{1}{2} N$

1个的 N 点DFT: $\frac{1}{2} N^2 + \frac{1}{2} N^2$

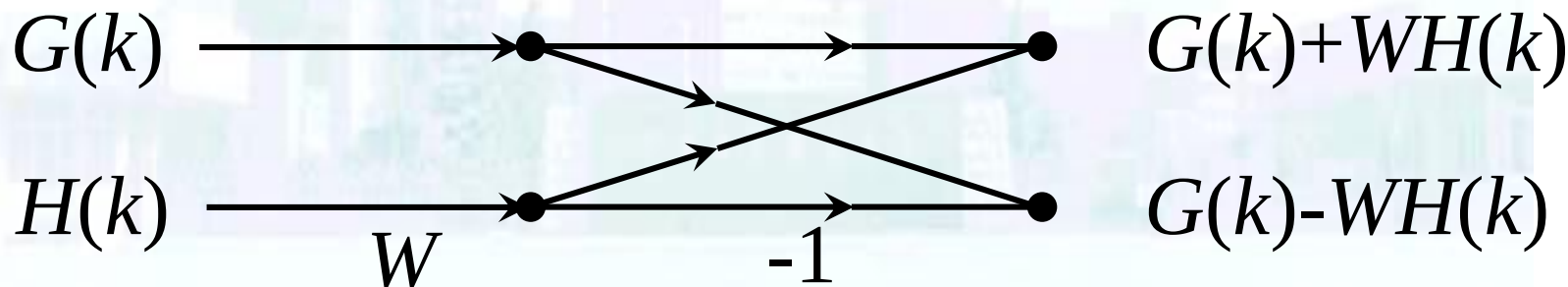
3.2 基2的DIT-FFT(按时间抽取的FFT)



3.2 基2的DIT-FFT(按时间抽取的FFT)

蝶形运算的定义

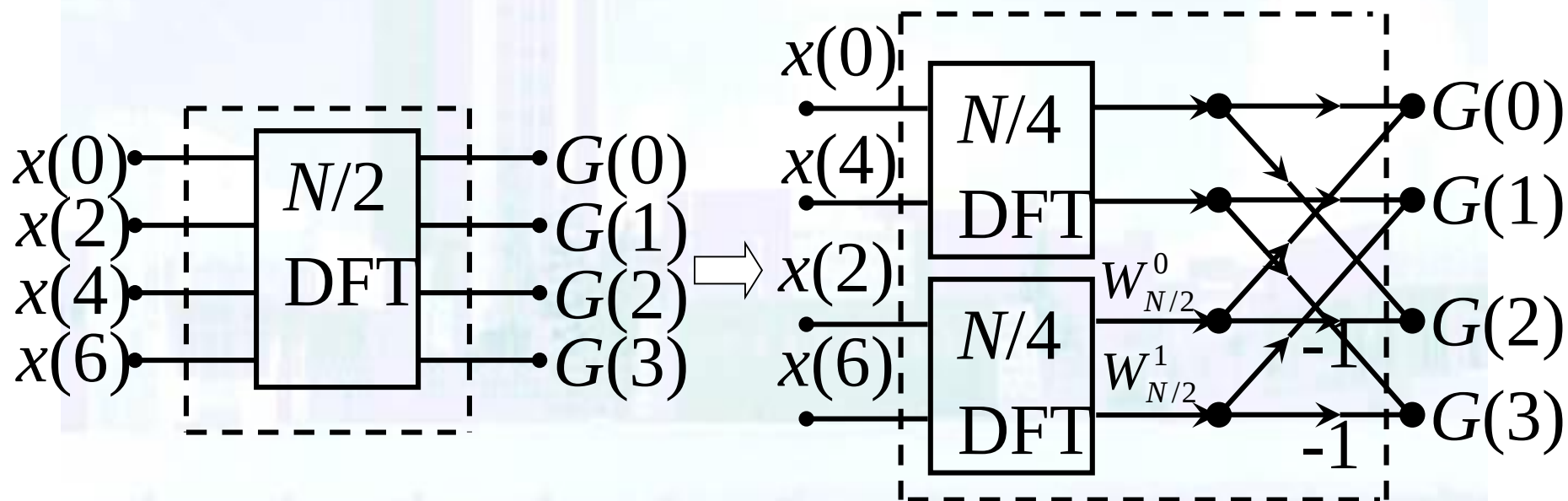
$$\left. \begin{aligned} X(k) &= G(k) + W_N^k H(k) \\ X(k + \frac{N}{2}) &= G(k) - W_N^k H(k) \end{aligned} \right\} k = 0, 1, 2, \dots, \frac{N}{2} - 1$$



3.2 基2的DIT-FFT(按时间抽取的FFT)

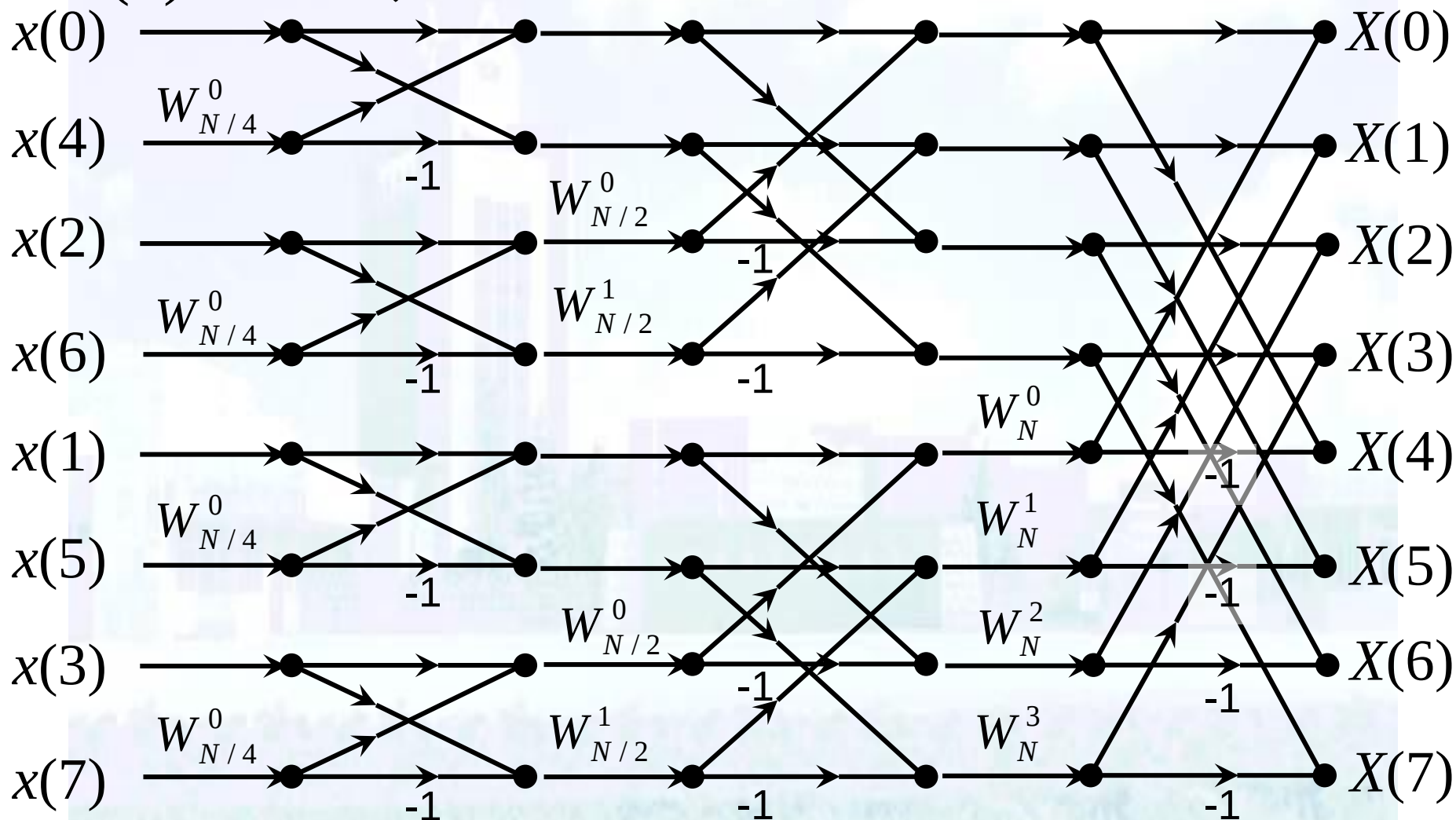
(2) $N/2$ 为偶数

$G(k)$ 和 $H(k)$ 的 $N/2$ 点DFT计算可分别继续分解为2个 $N/4$ 点DFT和蝶形运算。



3.2 基2的DIT-FFT(按时间抽取的FFT)

(3) $N=2^M$ ，分解过程可一直进行下去



3.2 基2的DIT-FFT(按时间抽取的FFT)

算法要点——乘法运算量

对 $N=2^M$ ，有 M 次递推，每次有 $N/2$ 个蝶形运算，

总共有 $\frac{1}{2}NM = \frac{1}{2}N\log_2 N$ 个蝶形运算。

1个蝶形包含1个乘法运算，总运算量为

$$\frac{1}{2}NM = \frac{1}{2}N\log_2 N \quad \text{次乘法。}$$

3.2 基2的DIT-FFT(按时间抽取的FFT)

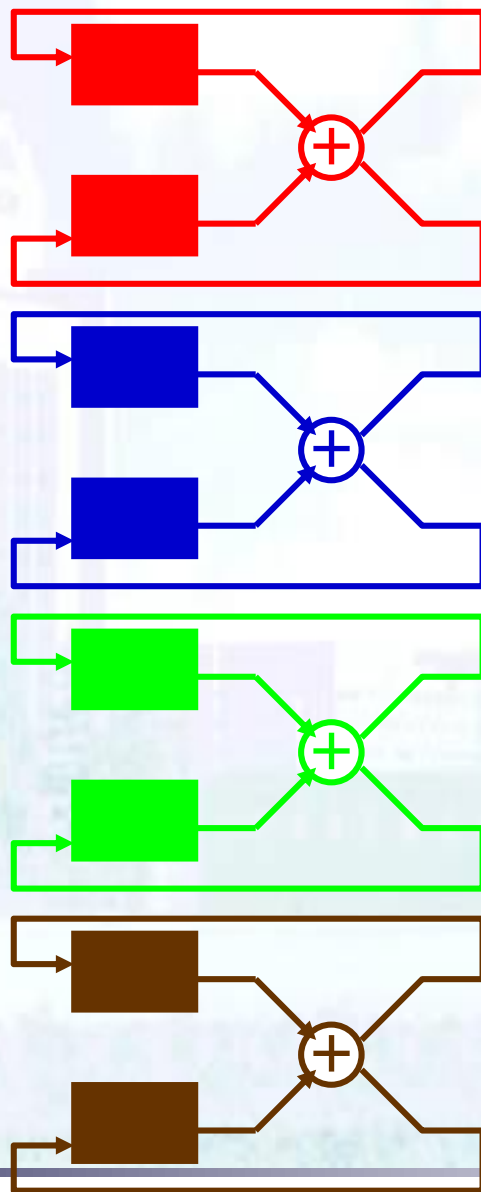
算法要点——原位（同址）运算

$$\begin{aligned}x_l(m) &= x_{l-1}(m) + W_N^P x_{l-1}(n) \\x_l(n) &= x_{l-1}(m) - W_N^P x_{l-1}(n)\end{aligned} \quad l = 1, \dots, M$$

输出数据存放在原来的输入数据存储单元，成为下一轮的输入数据，直至成为最后的计算结果。

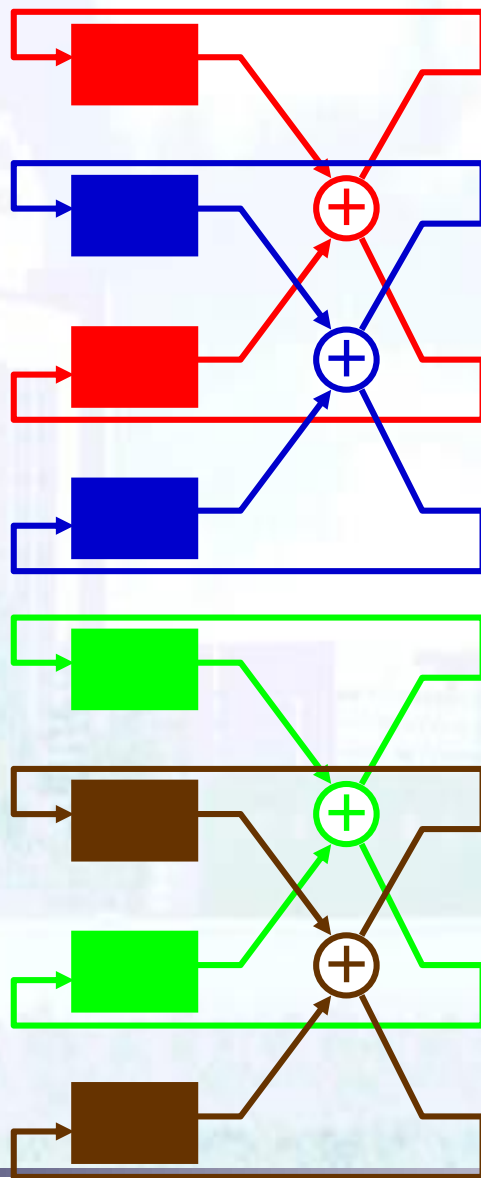
3.2 基2的DIT-FFT(按时间抽取的FFT)

$l=1$



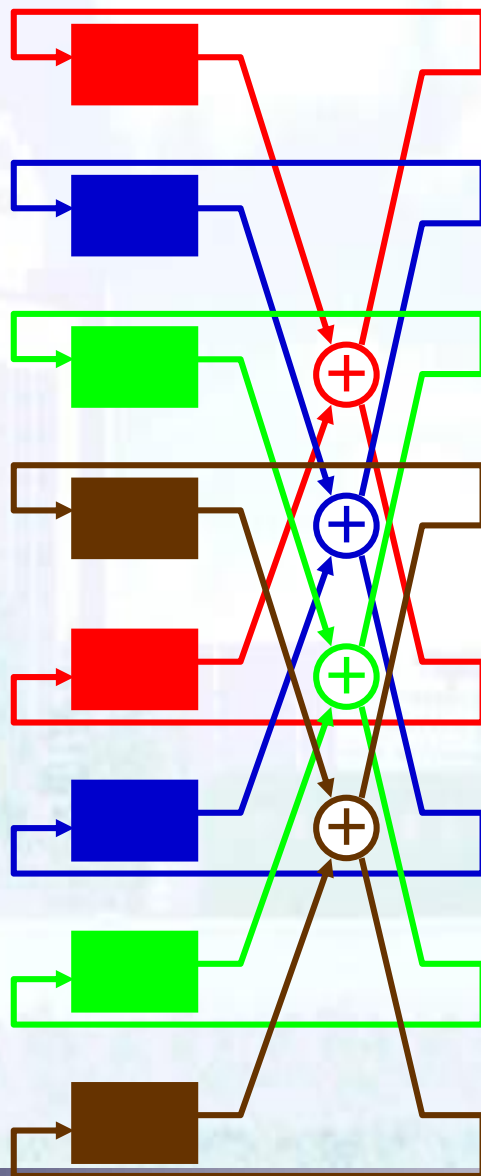
3.2 基2的DIT-FFT(按时间抽取的FFT)

$l=2$



3.2 基2的DIT-FFT(按时间抽取的FFT)

$l=3$

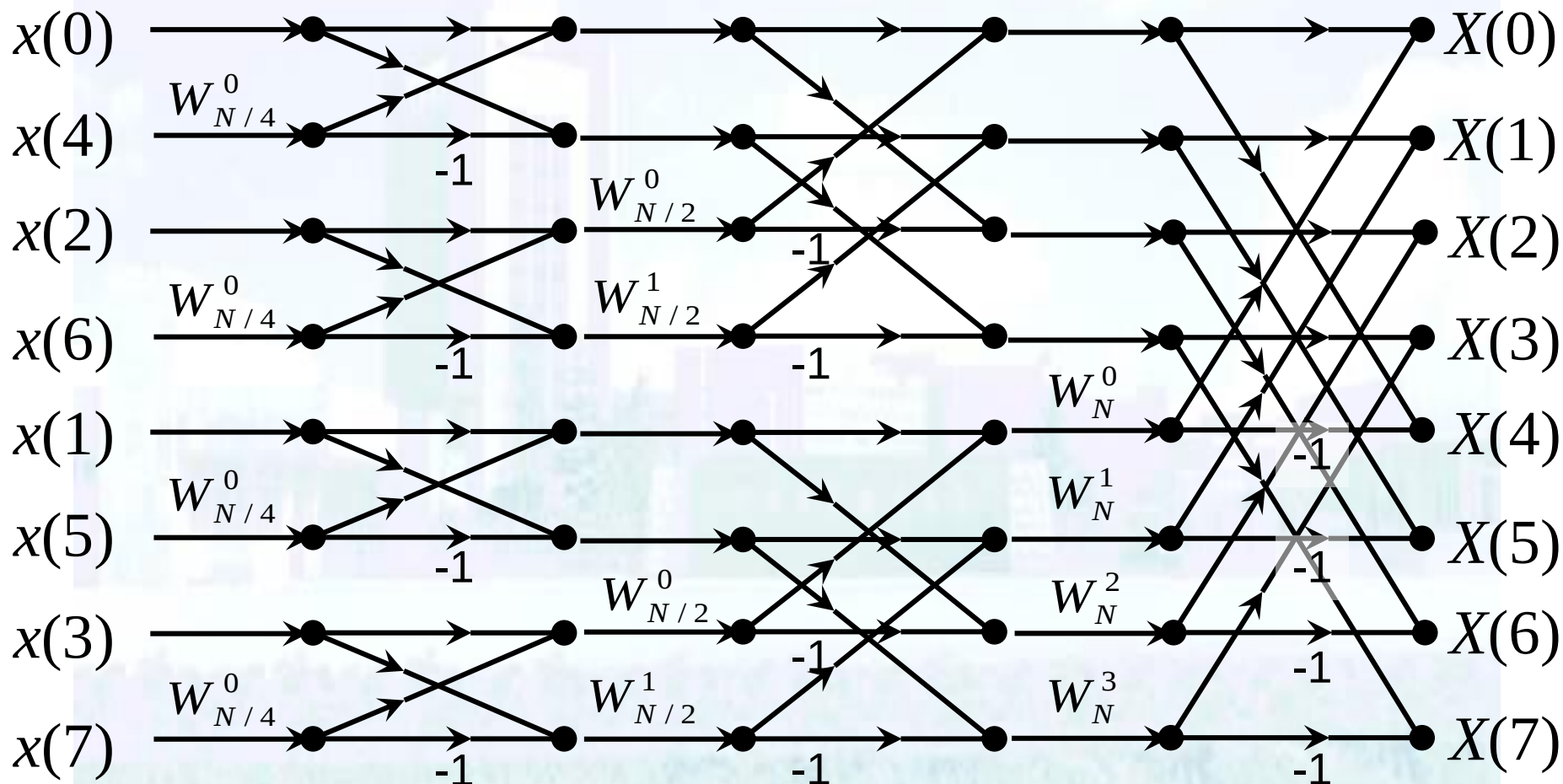


3.2 基2的DIT-FFT(按时间抽取的FFT)

算法要点——序数重排

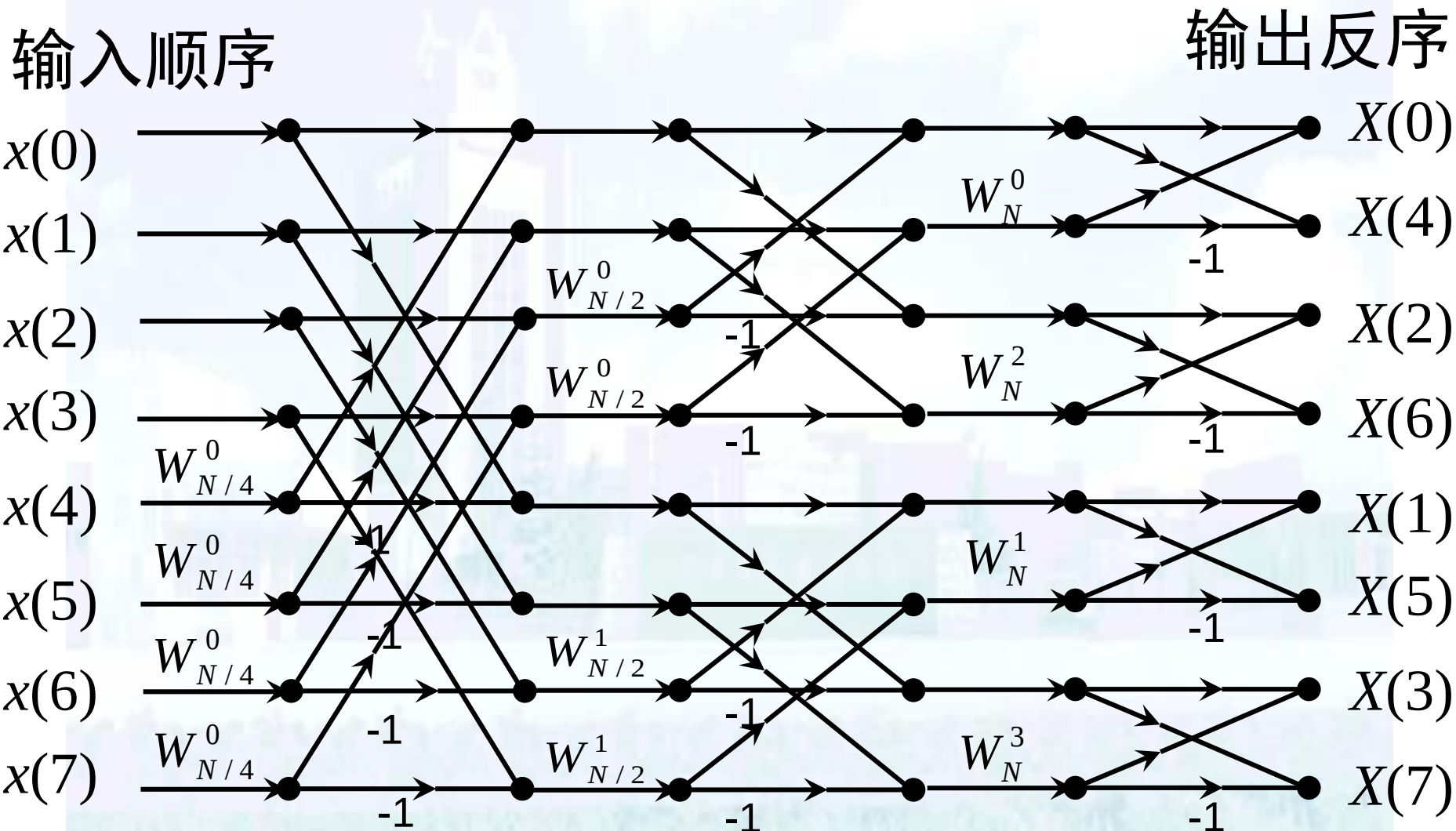
输入反序

输出顺序



3.2 基2的DIT-FFT(按时间抽取的FFT)

算法要点——序数重排



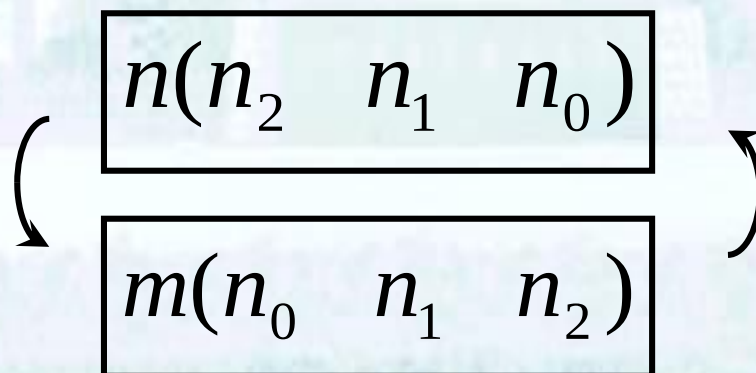
3.2 基2的DIT-FFT(按时间抽取的FFT)

反序规则

原序数 $n \Rightarrow$ 二进制 $n_2 n_1 n_0$

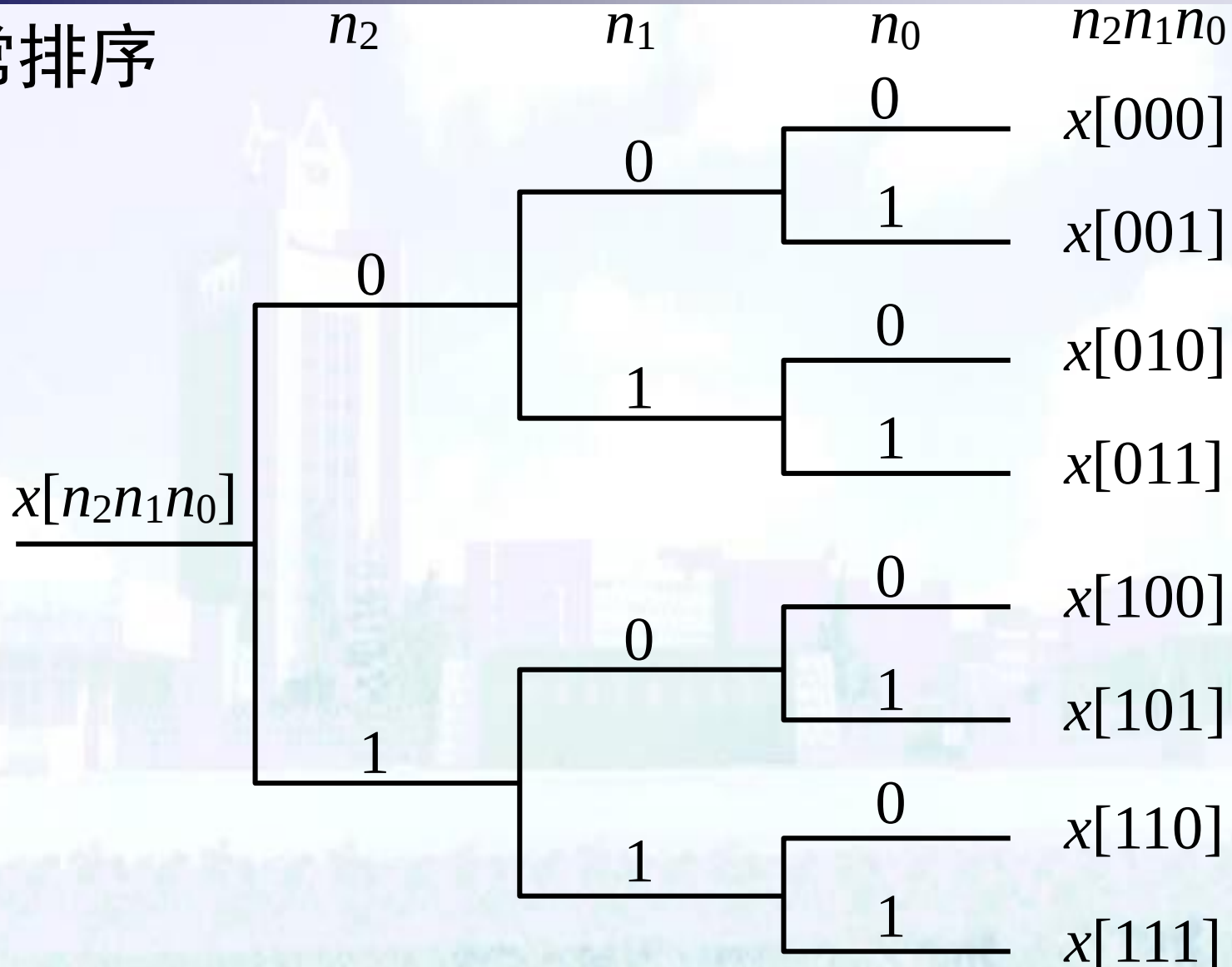
$\Rightarrow$ 反序二进制 $n_0 n_1 n_2 \Rightarrow$ 反序数 m

原序数为 n 的序列值放入第 m 个存储单元；
原序数为 m 的序列值放入第 n 个存储单元。

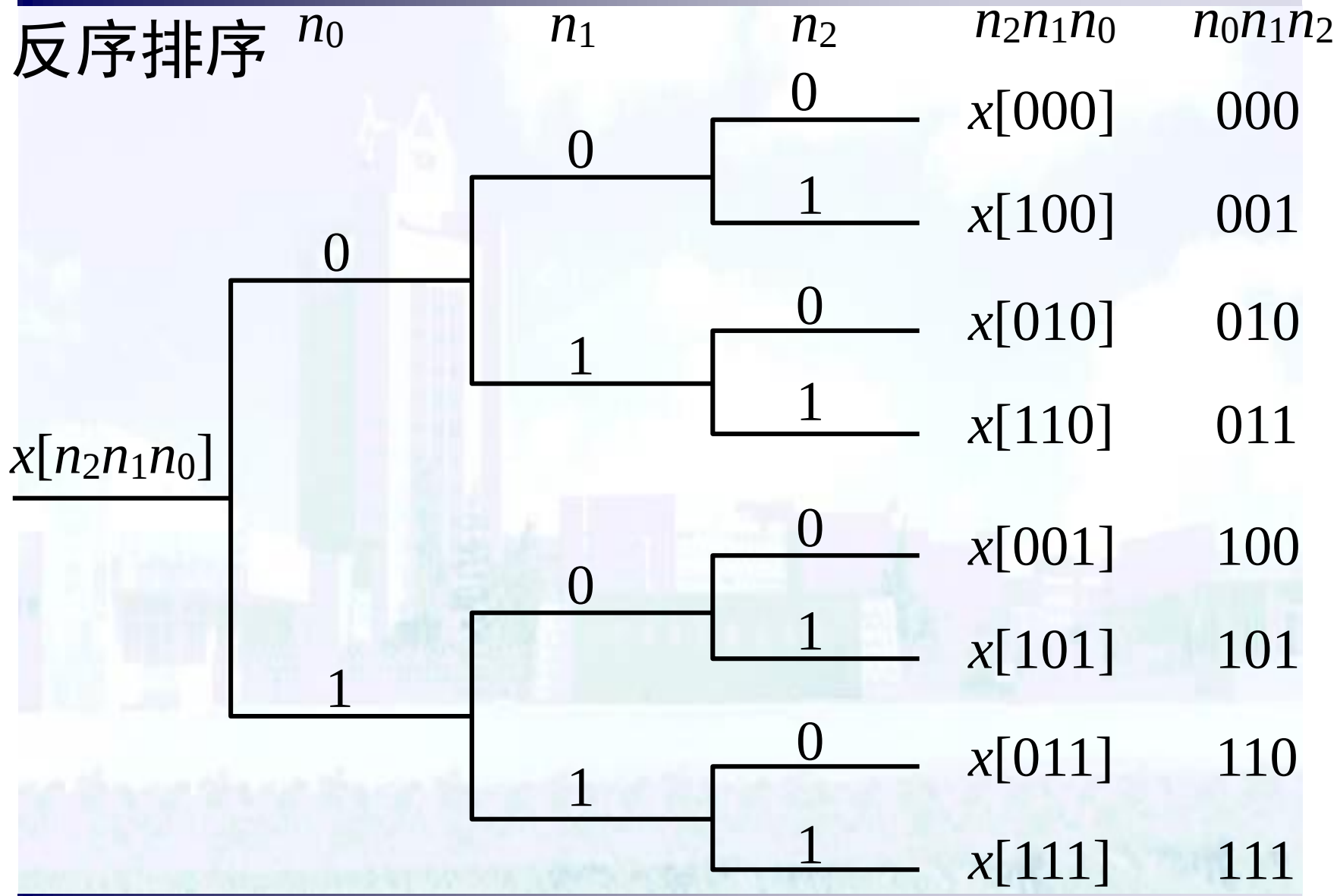


3.2 基2的DIT-FFT(按时间抽取的FFT)

正常排序



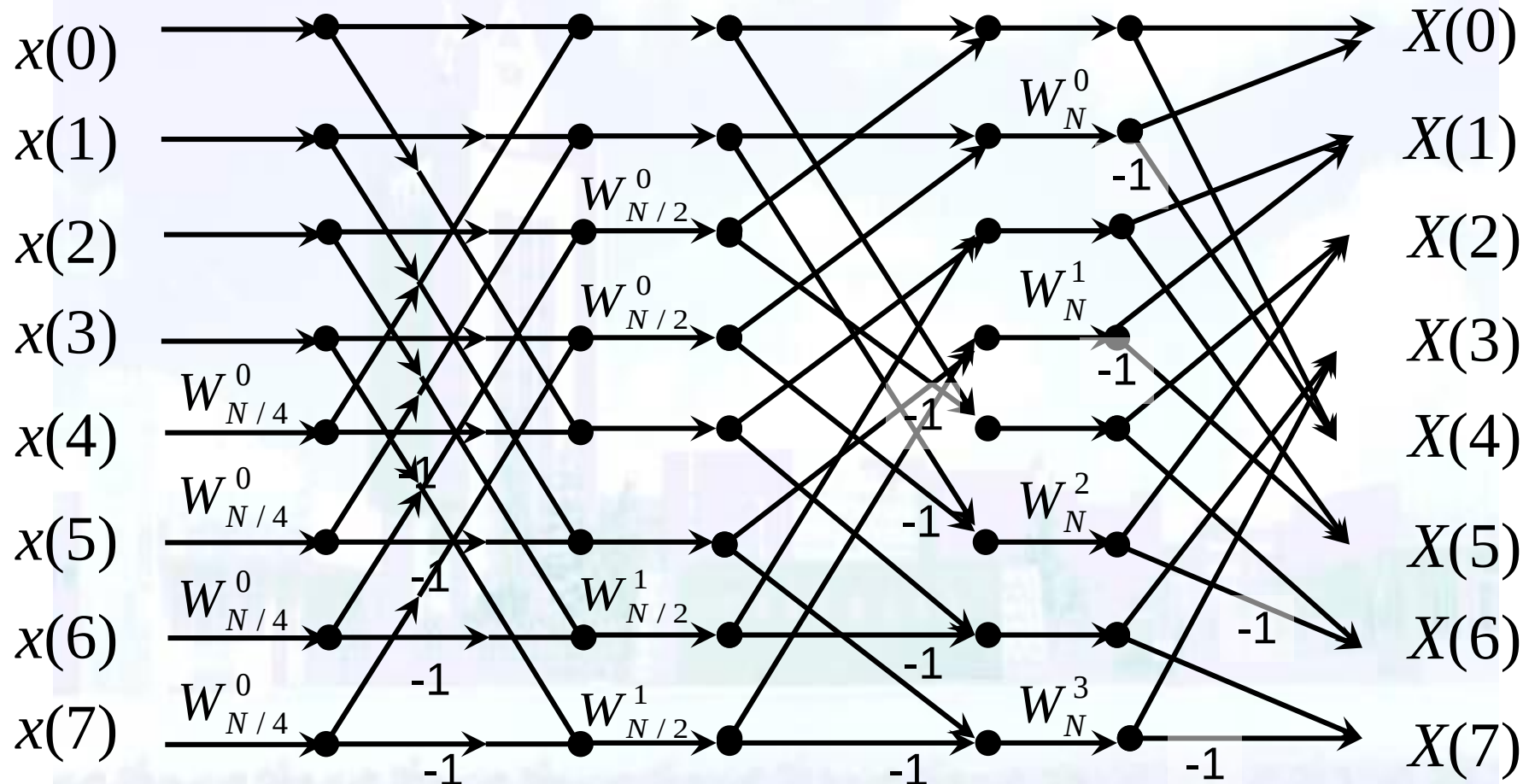
3.2 基2的DIT-FFT(按时间抽取的FFT)



3.2 基2的DIT-FFT(按时间抽取的FFT)

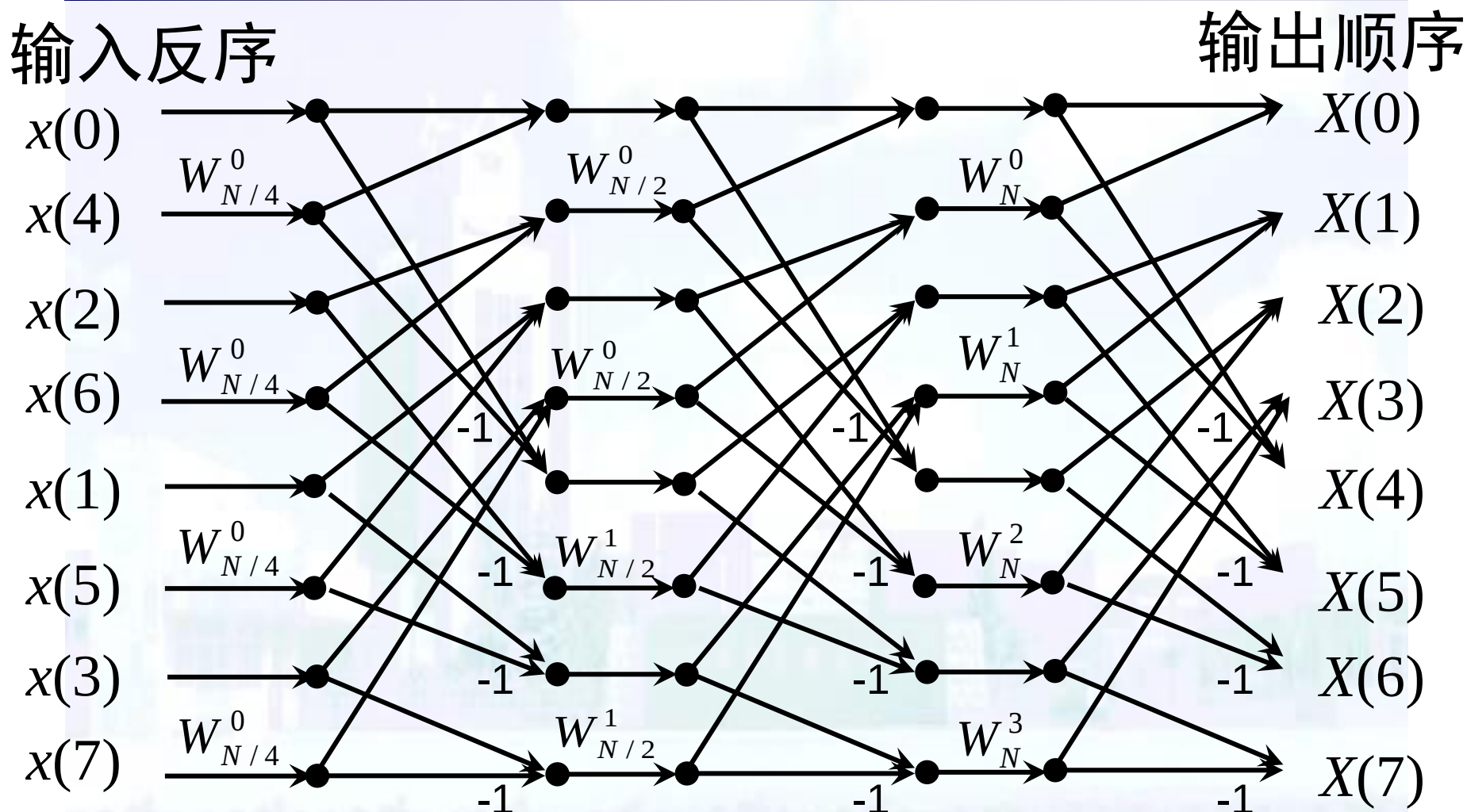
输入顺序

输出顺序



无法实现原位运算，需要增加存储器开销。

3.2 基2的DIT-FFT(按时间抽取的FFT)



具有相同的运算结构，非原位运算

3.3 基2的DIF-FFT(按频率抽取的FFT)

Decimation In Frequency

3.3 基2的DIF-FFT(按频率抽取的FFT)

N 为偶数时, $x(n)$ 按照前后顺序分为两部分

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

3.3 基2的DIF-FFT(按频率抽取的FFT)

注意到 $W_N^{Nk/2} = (-1)^k$

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

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

3.3 基2的DIF-FFT(按频率抽取的FFT)

将 $X(k)$ 按奇偶分别表达,

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

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

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

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

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

3.3 基2的DIF-FFT(按频率抽取的FFT)

偶数部分

$$X(2l) = \sum_{n=0}^{N/2-1} a_n W_{N/2}^{nl} \quad l=0,1,\dots, N/2 - 1$$

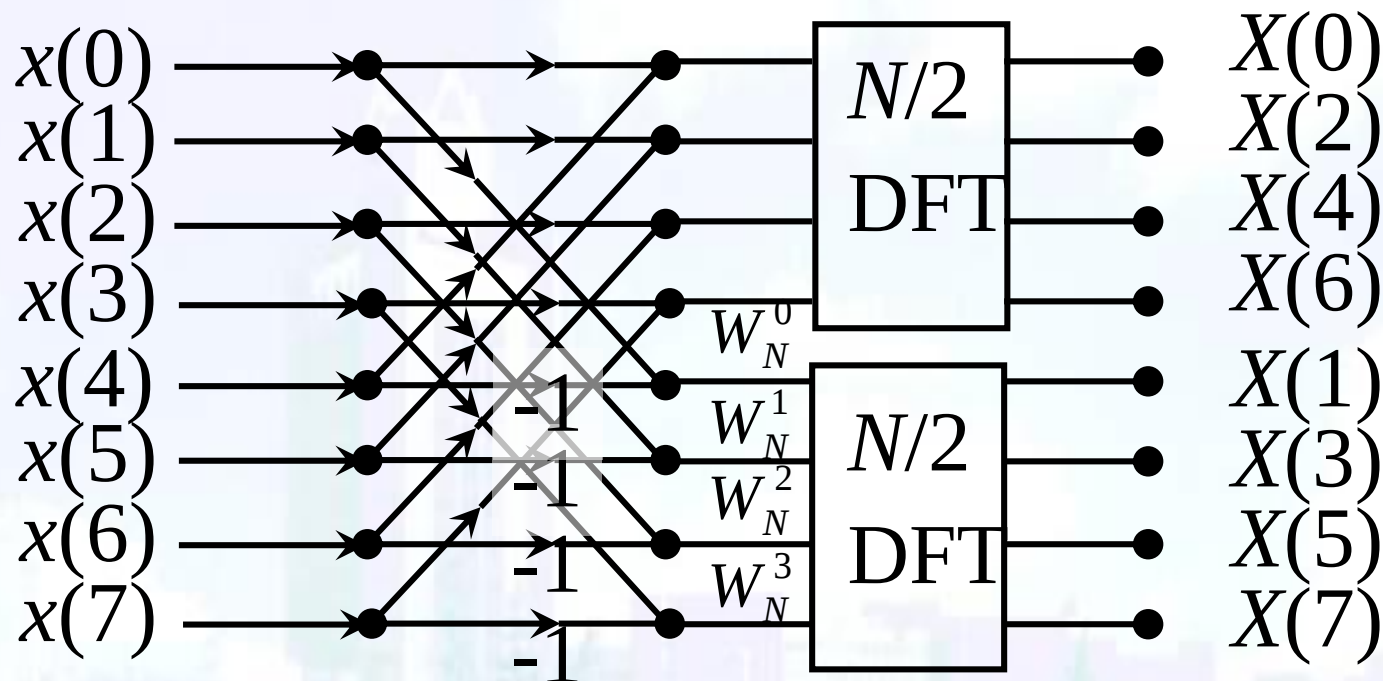
对序列 $a_n = x(n) + x(n + N/2)$ 的 $N/2$ 点 DFT

奇数部分

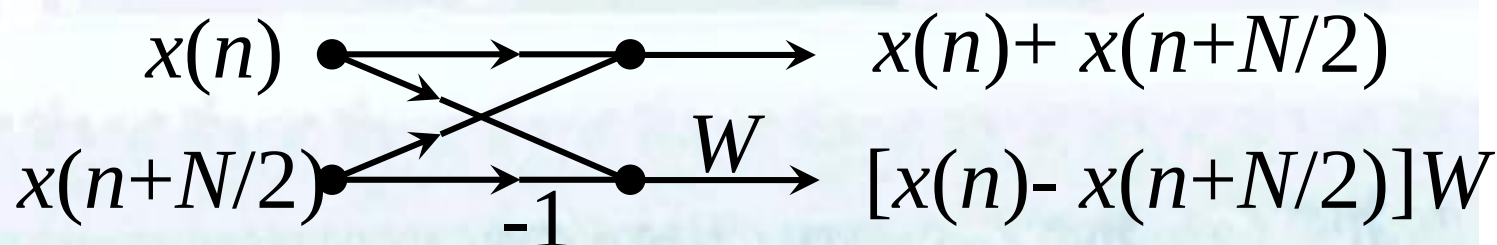
$$X(2l+1) = \sum_{n=0}^{N/2-1} b_n W_{N/2}^{nl} \quad l=0,1,\dots, N/2-1$$

对序列 $b_n = [x(n) - x(n + N/2)]W_N^n$ 的 $N/2$ 点 DFT

3.3 基2的DIF-FFT(按频率抽取的FFT)

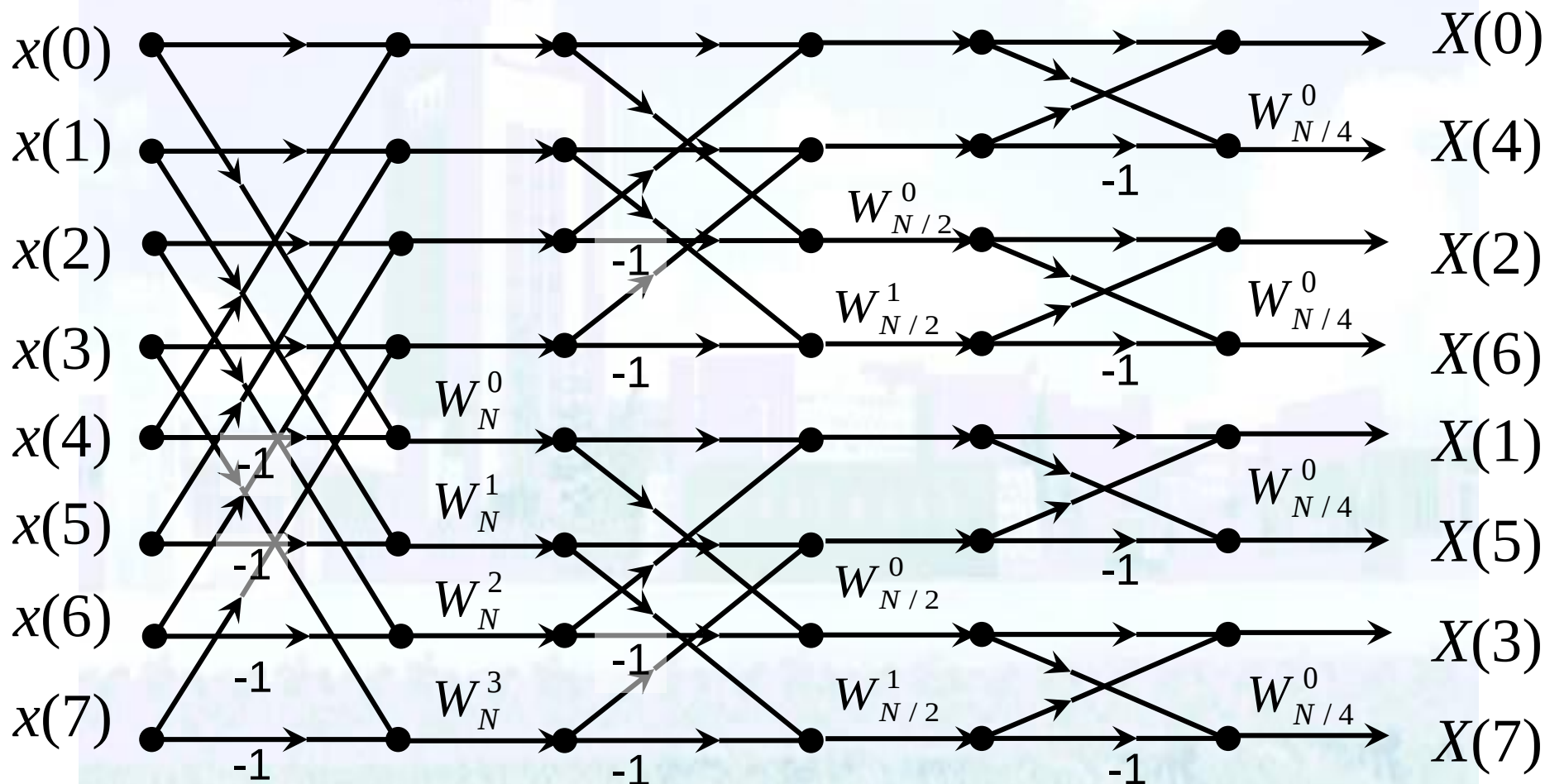


DIF蝶形运算的定义：



3.3 基2的DIF-FFT(按频率抽取的FFT)

对基2情形, $N=2^M$, $N/2$ 点DFT可按频率抽取原则进一步分解, 直至2点DFT



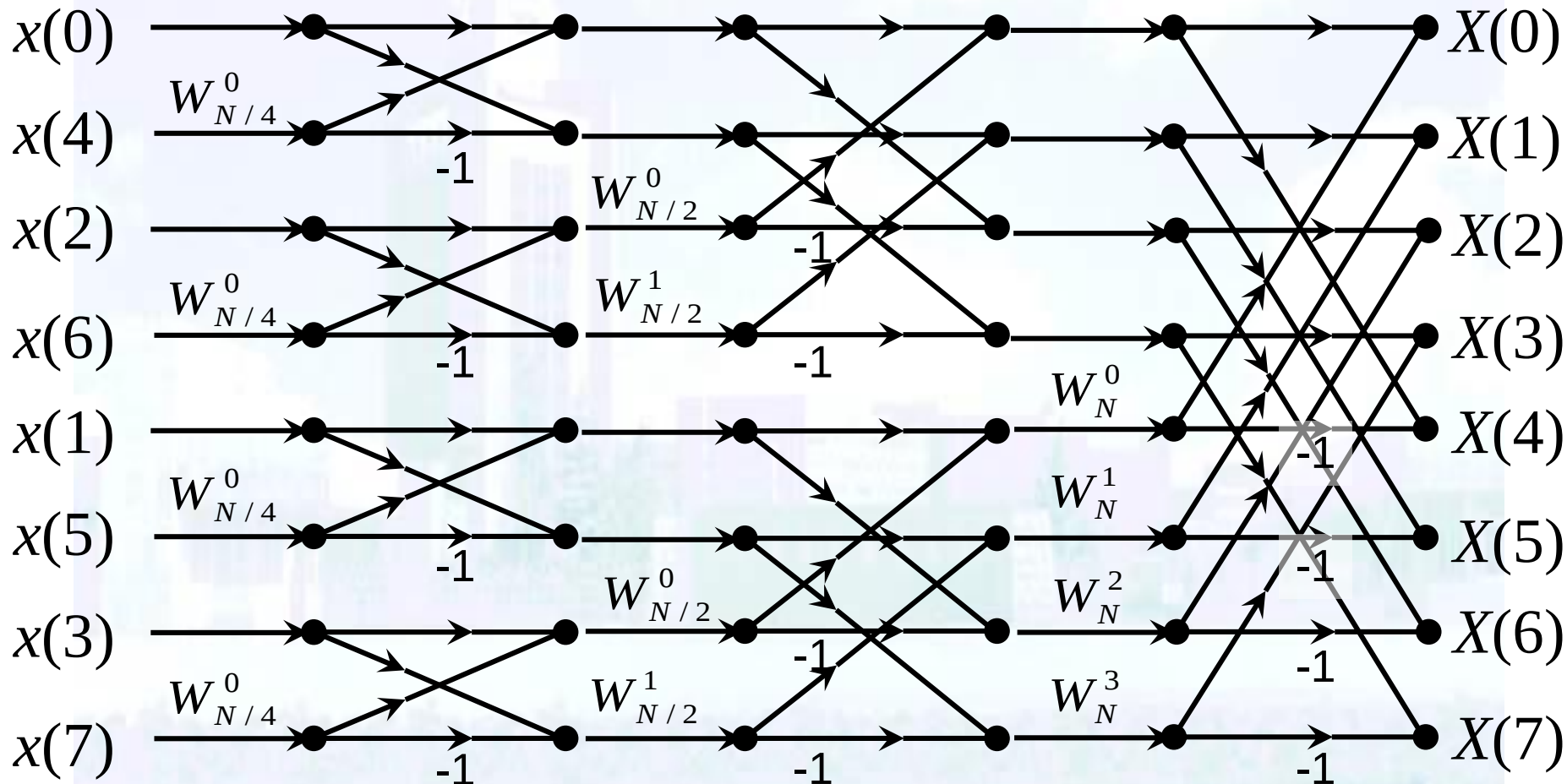
3.3 基2的DIF-FFT(按频率抽取的FFT)

算法要点

- 1)与DIT算法的特点基本相同，乘法运算量同为 $(N\log_2 N)/2$ ，蝶形运算同为原位运算。
- 2)DIF-FFT同样可以有变序输入，顺序输出。与DIT的区别不在于输入、输出序列的排序，而在于旋转因子所处的位置。
- 3)对DIT，将信号流程取反，即得DIF。同样，对DIF，将信号流程取反，即得DIT。

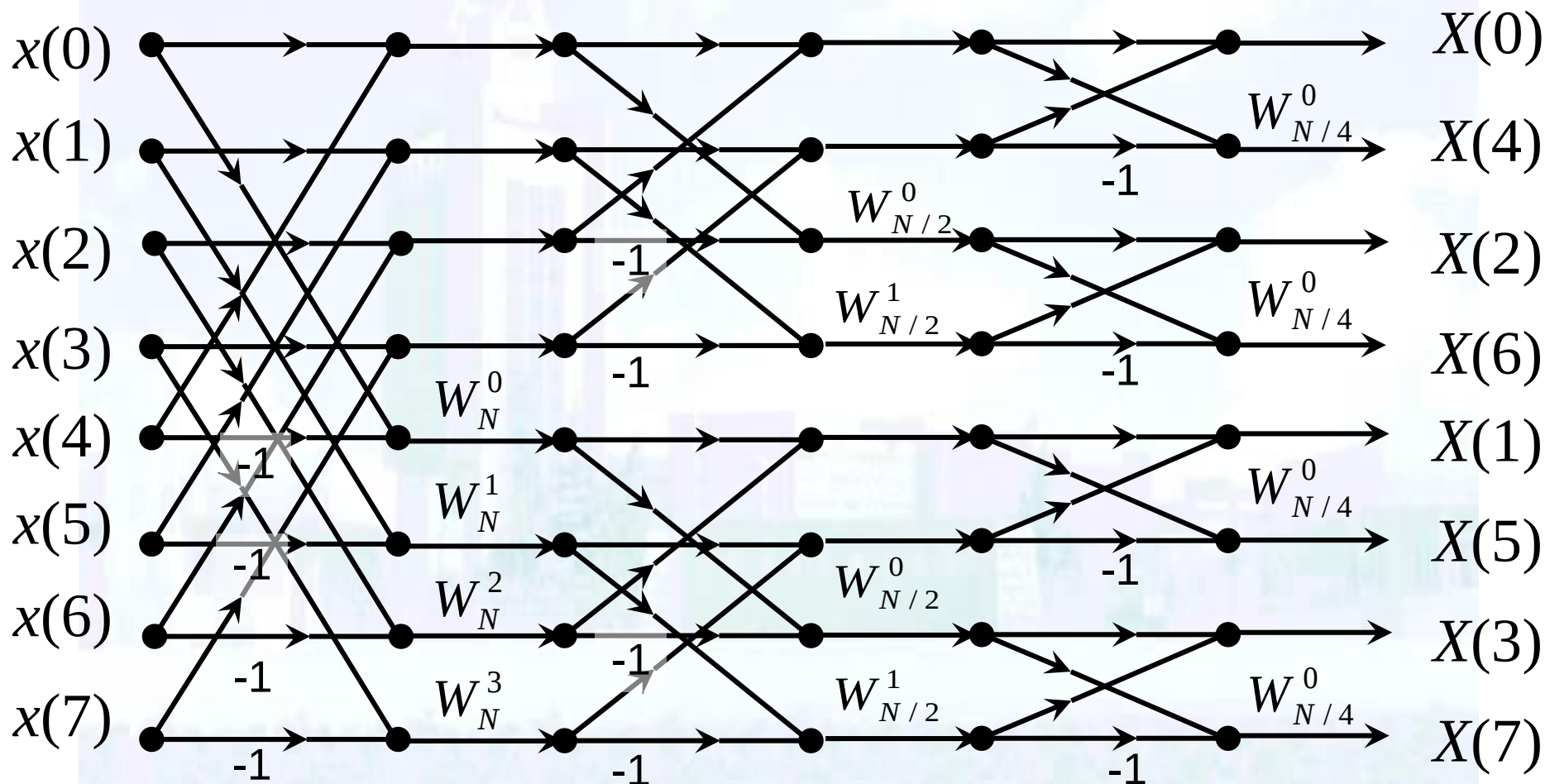
3.3 基2的DIF-FFT(按频率抽取的FFT)

DIT-FFT



3.3 基2的DIF-FFT(按频率抽取的FFT)

DIF-FFT





3.4 N 为复合数的FFT算法

3.4 N 为复合数的FFT算法

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

N 可以表达2个以上的整数相乘

设 $N = M \cdot L$

将 n 表达为 $n = n_1 M + n_0$

$$0 \leq n_1 \leq L-1, \quad 0 \leq n_0 \leq M-1$$

3.4 N 为复合数的FFT算法

$$\begin{aligned} X(k) &= \sum_{n_0=0}^{M-1} \sum_{n_1=0}^{L-1} x(n_1M + n_0) W_N^{k(n_1M + n_0)} \\ &= \sum_{n_0=0}^{M-1} W_N^{kn_0} \sum_{n_1=0}^{L-1} x(n_1M + n_0) W_N^{kn_1M} \\ &= \sum_{n_0=0}^{M-1} W_N^{kn_0} \sum_{n_1=0}^{L-1} x(n_1M + n_0) W_L^{kn_1} \end{aligned}$$

3.4 N 为复合数的FFT算法

将 k 表达为

$$k = k_1 L + k_0, \quad 0 \leq k_1 \leq M - 1, \quad 0 \leq k_0 \leq L - 1$$

利用关系 $W_N^{(k_1 L + k_0) n_0} = W_M^{k_1 n_0} \cdot W_N^{k_0 n_0}$

$$W_L^{(k_1 L + k_0) n_1} = W_L^{k_0 n_1}$$

得 $X(k_1 L + k_0)$

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

3.4 N 为复合数的FFT算法

1)定义

$$x_1(k_0, n_0) = \sum_{n_1=0}^{L-1} x(n_1M + n_0)W_L^{k_0n_1}$$

这是关于序列

$$x(n_1M + n_0), \quad n_1 = 0, \dots, L-1$$

的一个标准的 L 点DFT。

n_0 有 M 个不同取值，共有 M 个 L 点的DFT。

3.4 N 为复合数的FFT算法

2)定义 $x_2(k_0, n_0) = x_1(k_0, n_0)W_N^{k_0 n_0}$

$$\text{则 } X(k_1 L + k_0) = \sum_{n_0=0}^{M-1} x_2(k_0, n_0) W_M^{k_1 n_0}$$

这是关于序列

$$x_2(k_0, n_0), \quad n_0 = 0, \dots, M-1$$

的 M 点DFT。

不同的 k_0 ，对应着不同的序列。 k_0 有 L 个不同取值，共有 L 个 M 点DFT。

3.4 N 为复合数的FFT算法

所以, $X(k)$ 可分解为:

M 个 L 点DFT

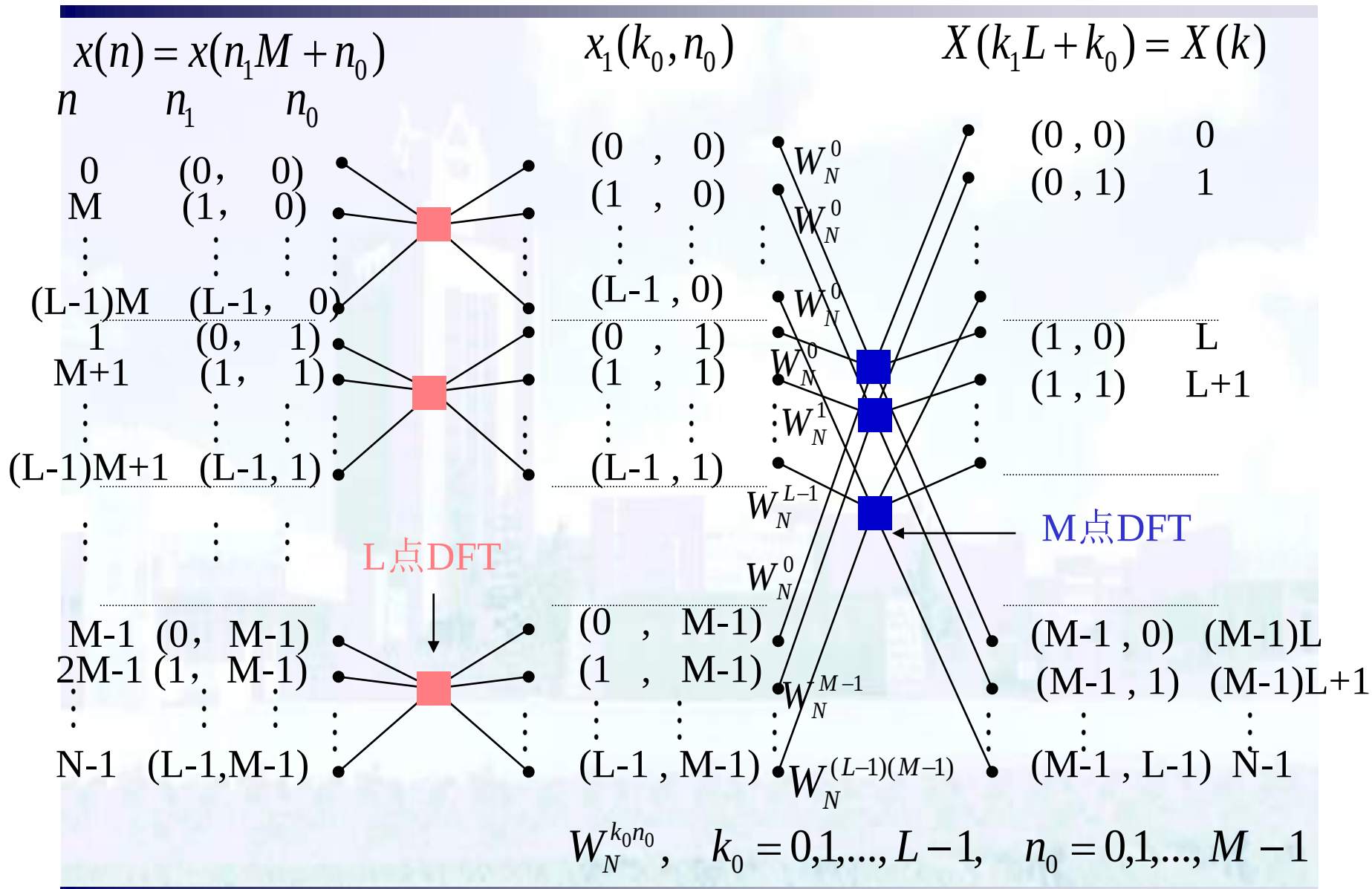
L 个 M 点DFT

中间结果的 N 次复数乘法

乘法运算量:

$$M \cdot L^2 + L \cdot M^2 + N = N(L + M + 1)$$

3.4 N 为复合数的FFT算法



3.4 N 为复合数的FFT算法

关于求和次序

在 $n = n_1M + n_0$, $k = k_1L + k_0$,

$$0 \leq n_1, k_0 \leq L - 1, \quad 0 \leq n_0, k_1 \leq M - 1$$

的定义下，表达是唯一的：

$$X(k_1L + k_0)$$

$$= \sum_{n_0=0}^{M-1} [W_N^{k_0n_0} \sum_{n_1=0}^{L-1} x(n_1M + n_0) W_L^{k_0n_1}] W_M^{k_1n_0}$$

3.4 N 为复合数的FFT算法

关于式(4-38)

$$X(k_1L + k_0)$$

$$= \sum_{n_1=0}^{L-1} \sum_{n_0=0}^{M-1} x(n_1, n_0) W_N^{(k_1L + k_0)(n_1M + n_0)}$$

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

3.4 N 为复合数的FFT算法

中间变量

$$x_1(n_1, n_0, k_0) = x(n_1, n_0) W_N^{k_0 n_0}$$

则：

$$x_2(n_1, k_0, k_1) = \sum_{n_0=0}^{M-1} x_1(n_1, n_0, k_0) W_M^{k_1 n_0}$$

这是一个标准的 M 点DFT。

3.4 N 为复合数的FFT算法

但是

$$X(k_0, k_1) = \sum_{n_1=0}^{L-1} x_2(n_1, k_0, k_1) W_L^{k_0 n_1}$$

不是一个标准的DFT

序列 $x_2(n_1, k_0, k_1)$ 本身是 k_0 的函数，随着 k_0 的变化而变化

即 $X(k) = \sum x(n, k) W_N^{kn}$ 不是DFT。

3.4 N 为复合数的FFT算法

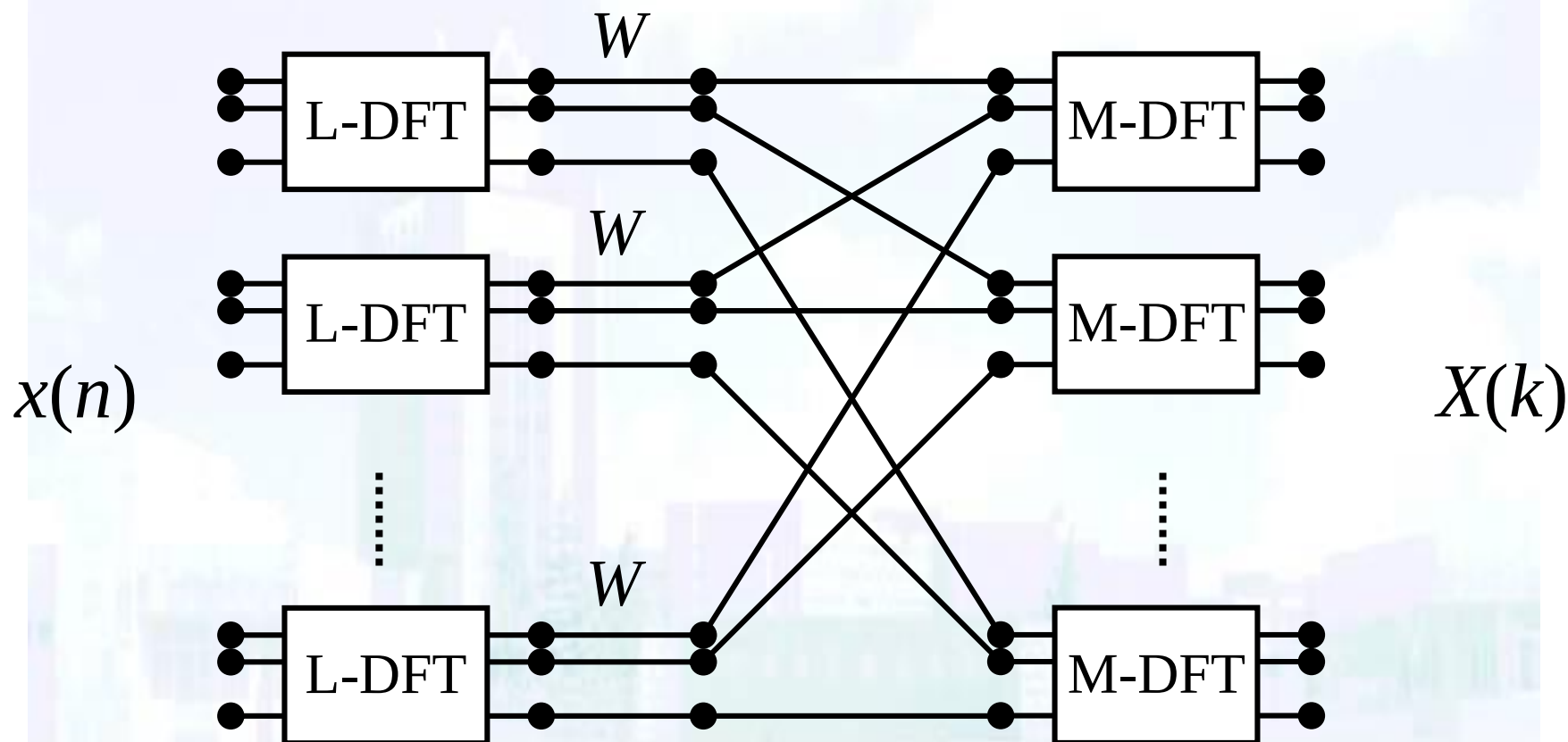
当 $n = n_1M + n_0$, $k = k_1L + k_0$ 时

求和顺序唯一：

$$\begin{aligned} & X(k_1L + k_0) \\ &= \sum_{n_0=0}^{M-1} [W_N^{k_0n_0} \sum_{n_1=0}^{L-1} x(n_1M + n_0) W_L^{k_0n_1}] W_M^{k_1n_0} \end{aligned}$$

3.4 N 为复合数的FFT算法

关于时间抽取与频率抽取



$N=8$: 取 $L=4$, $M=2$, 时间抽取
取 $L=2$, $M=4$, 频率抽取

3.4 N 为复合数的FFT算法

$$1) L=4, M=2$$

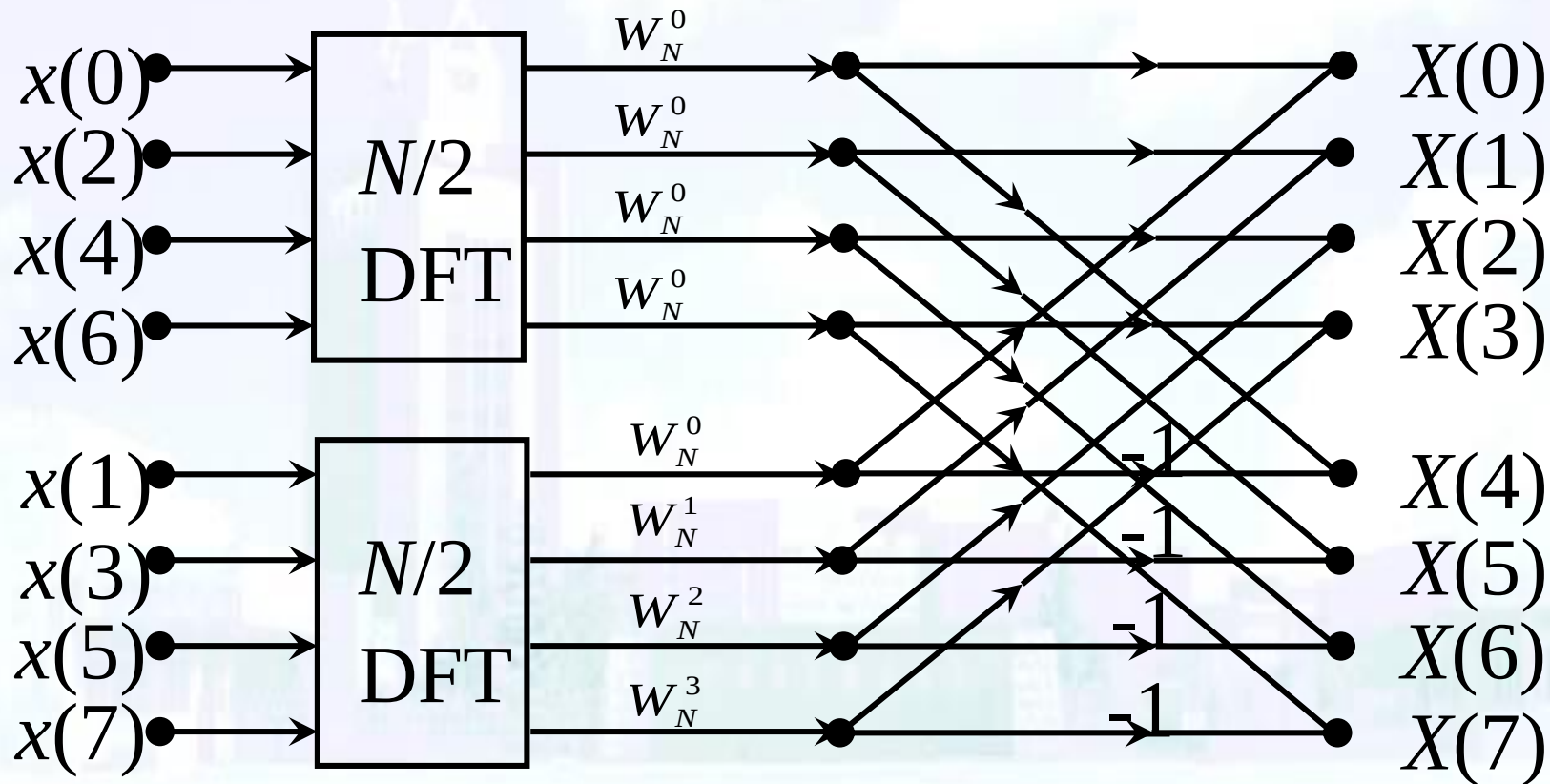
$$X(4k_1 + k_0)$$

$$= \sum_{n_0=0}^1 [W_8^{k_0 n_0} \sum_{n_1=0}^3 x(2n_1 + n_0) W_4^{k_0 n_1}] W_2^{k_1 n_0}$$

$$W_N^{k_0 n_0}, \quad k_0 = 0, 1, 2, 3 \quad n_0 = 0, 1$$

所谓按时间抽取

3.4 N 为复合数的FFT算法



3.4 N 为复合数的FFT算法

$$2) L=2, M=4$$

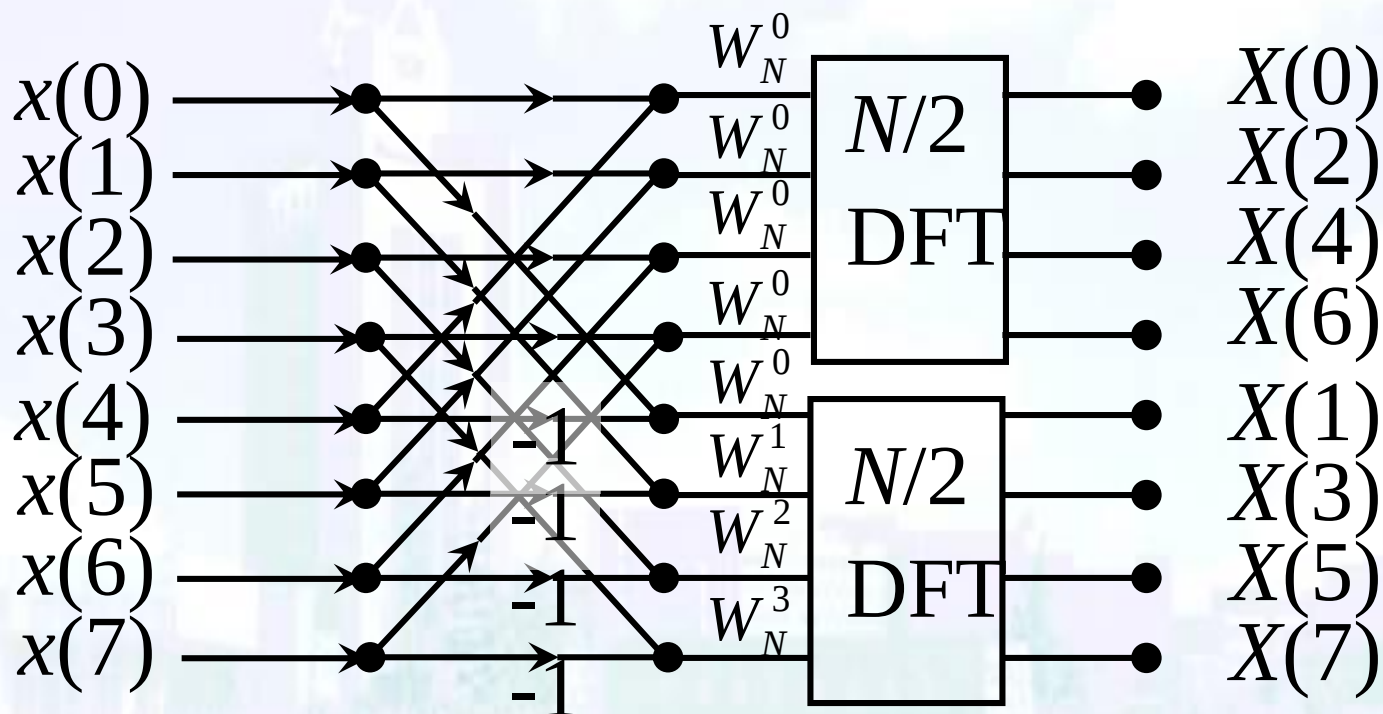
$$X(2k_1 + k_0)$$

$$= \sum_{n_0=0}^3 [W_8^{k_0 n_0} \sum_{n_1=0}^1 x(4n_1 + n_0) W_2^{k_0 n_1}] W_4^{k_1 n_0}$$

$$W_N^{k_0 n_0}, \quad n_0 = 0, 1, 2, 3, \quad k_0 = 0, 1$$

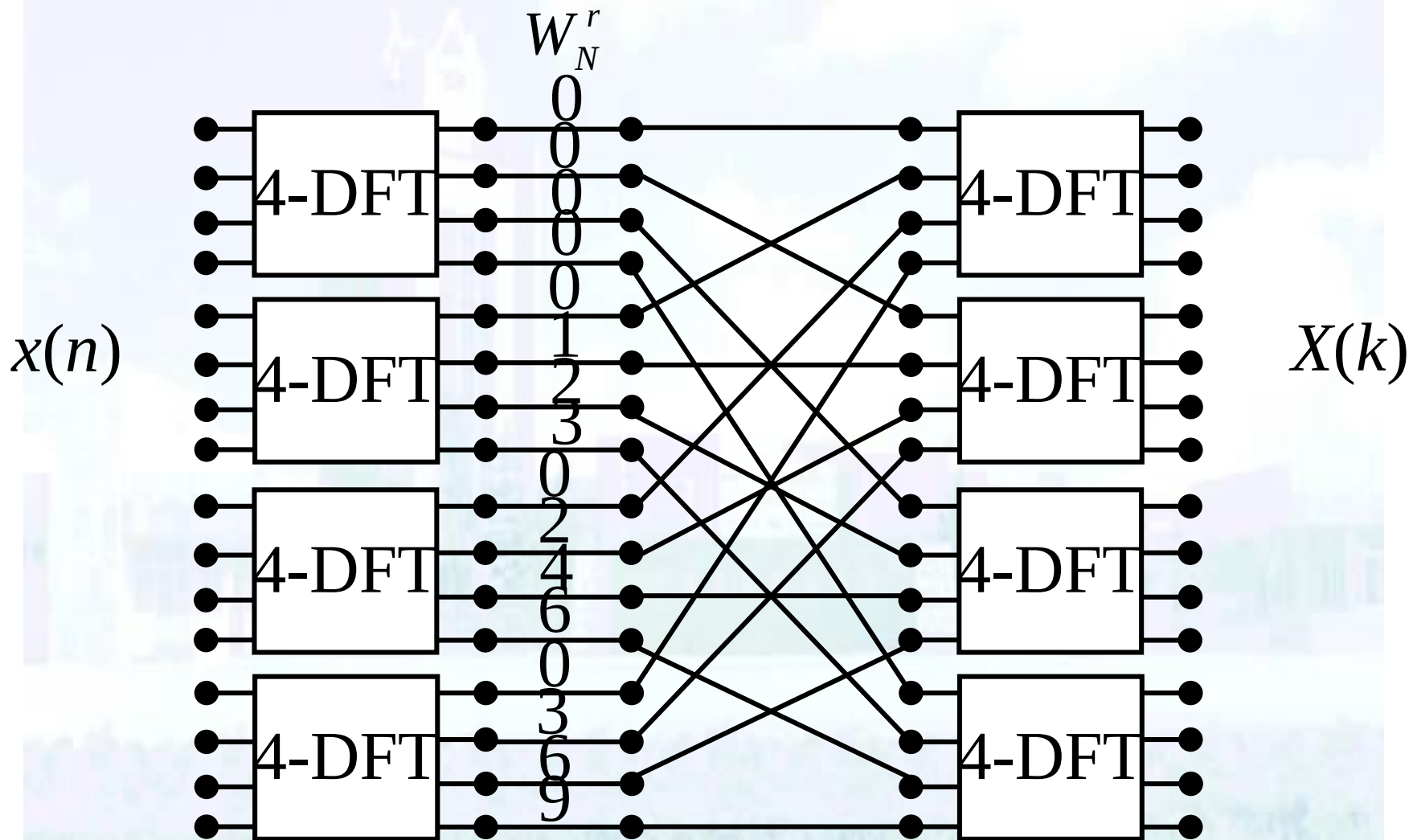
所谓按频率抽取

3.4 N 为复合数的FFT算法



3.4 N 为复合数的FFT算法

关于基4的快速算法 $N=4^M$ 以 $N=16$ 为例



3.4 N 为复合数的FFT算法

对 $N=4^M$

共有 M 级4点DFT

4点DFT本身没有乘法运算

每级有 $N/4$ 个基4蝶形

每个蝶形的系数乘法为3次

每级 $3N/4$ 次系数乘法

$$\frac{3}{4} NM = \frac{3}{4} N \cdot \frac{1}{2} \log_2 N = \frac{3}{8} N \log_2 N$$

3.4 N 为复合数的FFT算法

一般地, $N=r^M$

最小DFT运算单位为 r -DFT, 基 r 算法

第一步分解:

$$\begin{aligned} X(k_1, k_0) \\ = \sum_{n_0=0}^{\frac{N}{r}-1} [W_N^{k_0 n_0} \sum_{n_1=0}^{r-1} x(n_1, n_0) W_r^{k_0 n_1}] W_{N/r}^{k_1 n_0} \end{aligned}$$

3.4 N 为复合数的FFT算法

若把每个 r -DFT 看成1个运算单元，则整个分解共有 M 级运算，每级有 N/r 个运算单元

设每个 r -DFT 的 **实际** 乘法运算量为 N_r ，系数乘法 $(r-1)$ 次，总运算量为

$$\frac{N}{r} (N_r + r - 1) \cdot M$$

3.4 N 为复合数的FFT算法

基2:

$$r = 2, \quad N_r = 0, \quad M = \log_2 N \Rightarrow \frac{1}{2} N \log_2 N$$

基4:

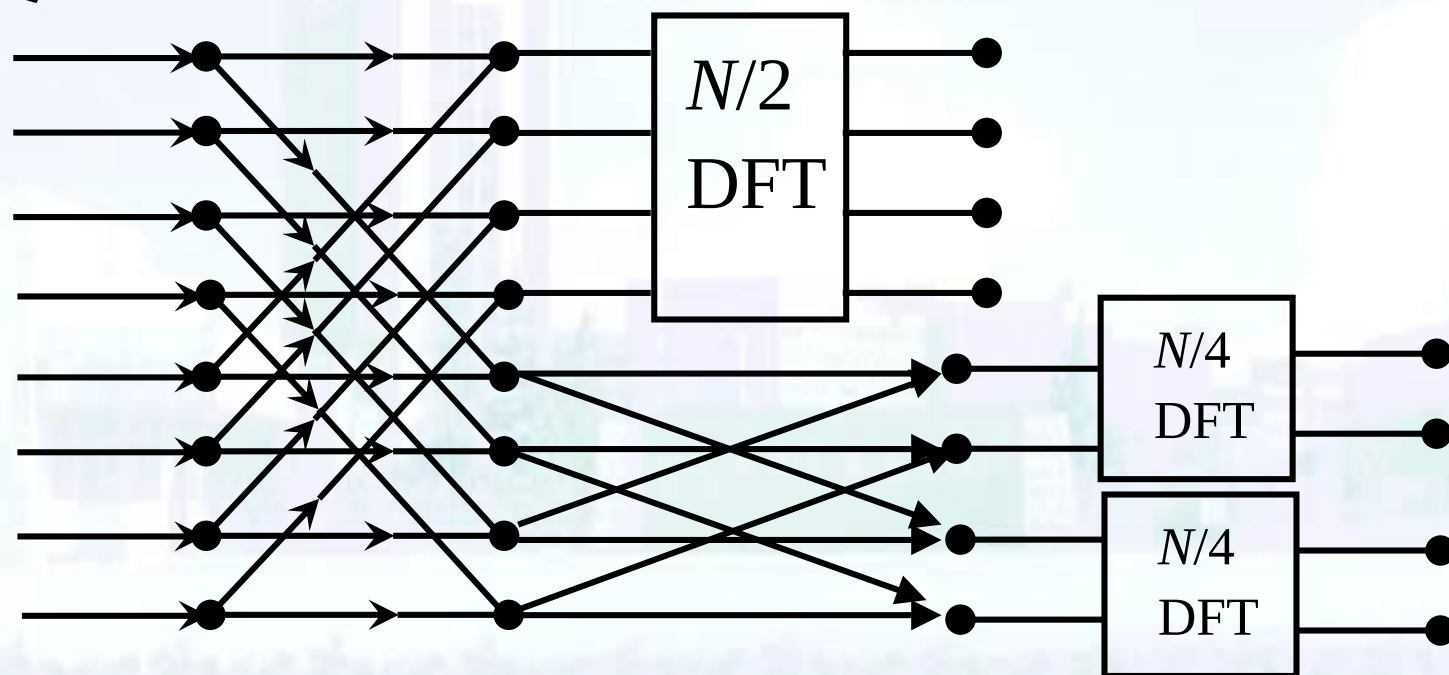
$$r = 4, \quad N_r = 0, \quad M = \log_4 N \Rightarrow \frac{3}{8} N \log_2 N$$

3.4 N 为复合数的FFT算法

分裂基算法（1984年）

思路：减少变换的系数乘法

设 $N=4P$



若 $N/2=4Q$ ， $N/4=4R$ ，则可进一步分解， L 蝶形

3.4 N 为复合数的FFT算法

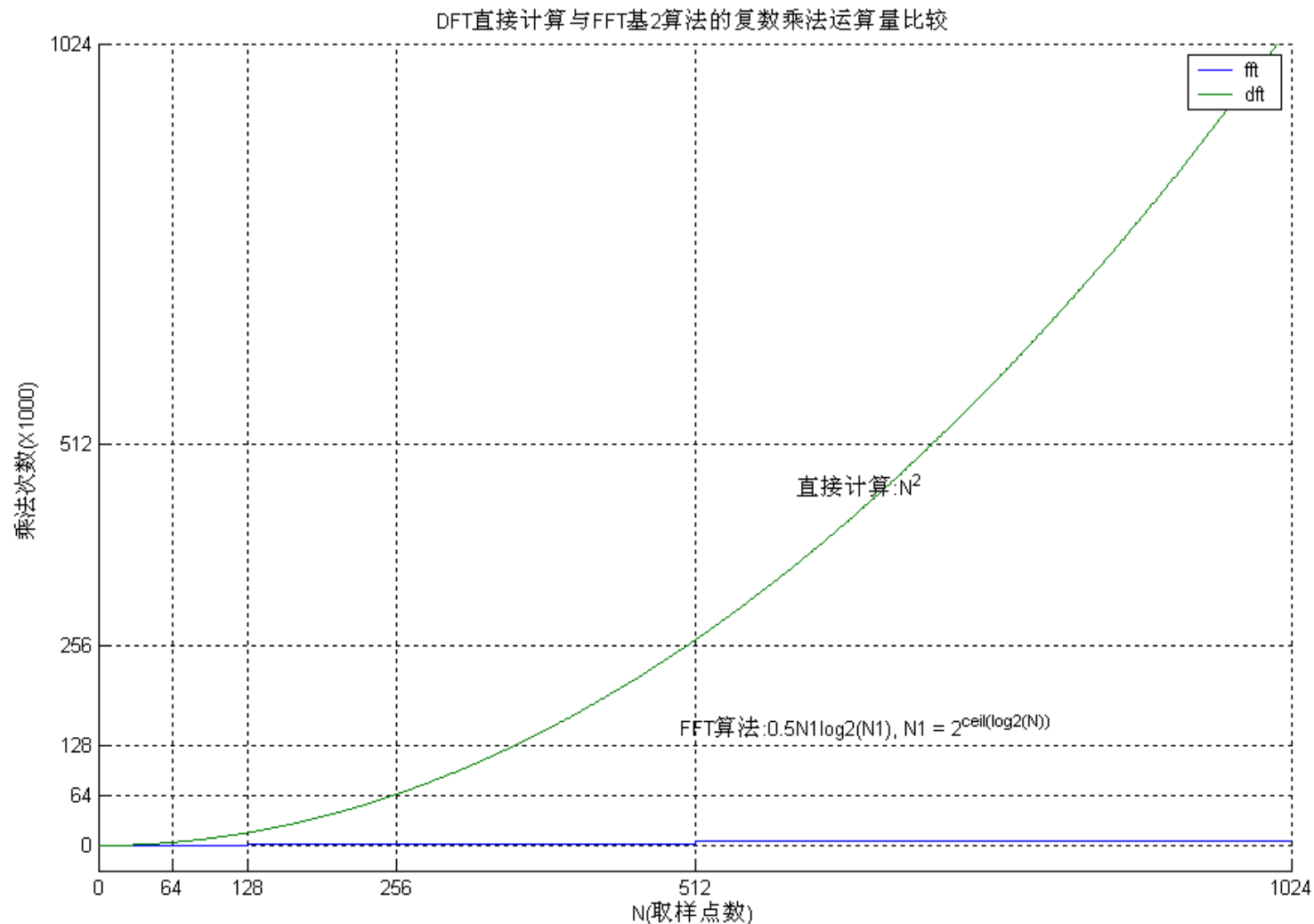
分裂基算法

若 $N=2^M$ ，则完全分解后，复系数乘法运算量约为

$$\frac{1}{3} N \log_2 N$$

乘法运算量比较

基2	$\frac{1}{2} N \log_2 N$	1
基4	$\frac{3}{8} N \log_2 N$	0.75
分裂基	$\frac{1}{3} N \log_2 N$	0.67



3.4 N 为复合数的FFT算法

关于IDFT的快速计算

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

3.4 N 为复合数的FFT算法

1)将系数 W_N^{nk} 换成 W_N^{-nk} ，并将最后结果乘以常数 $1/N$ ，按照上述两种方法编制FFT程序进行IDFT运算

2)使用常规FFT程序，对 $X(k)$ 取共轭，

$$x(n) = \frac{1}{N} \{DFT[X^*(k)]\}^*$$

3)使用常规FFT程序，在圆周移位定义下倒置

$$x(N - n) = \frac{1}{N} DFT[X(k)]$$



FFT算法的一般要点

- 将一个长的DFT分解为多个短的DFT
- 分解算法用标准的DFT计算模块表达
- 分层处理，循环分解至最小计算模块
- 乘法运算的来源，分解过程的系数修正
- 算法优化目标，减少系数乘法次数



3.5 线性调频Z变换 (CZT)

3.5 线性调频Z变换 (CZT)

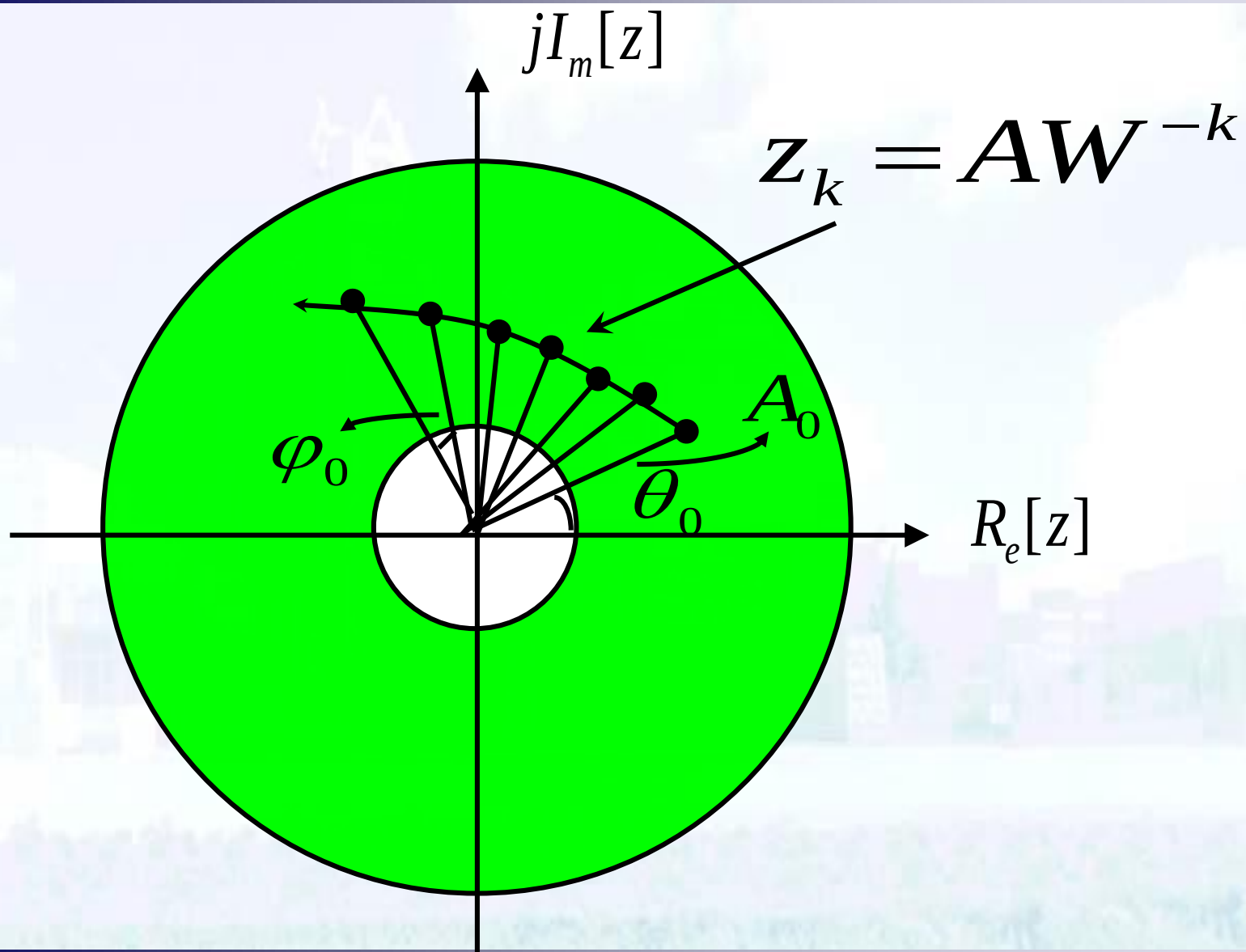
DFT: 对Z变换 $X(z)$ 在单位圆上的一种等间距取样

$$\begin{aligned} DFT[x(n)] &= X(z_k) \Big|_{z_k = e^{j\frac{2\pi}{N}k}} \\ &= \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}nk} \end{aligned}$$

CZT: 对Z变换 $X(z)$ 更一般的一种取样方式:

- (1) 不限于单位圆上, 而是一个螺旋线
- (2) 取样点数为 M , 可不等于 N
- (3) 取样的起点、范围任意

3.5 线性调频Z变换 (CZT)



3.5 线性调频Z变换 (CZT)

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

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

$$A = A_0 e^{j\theta_0} \quad \text{-----起始点}$$

$$W = W_0 e^{-j\varphi_0} \quad \text{-----控制旋进的方向和步进大小}$$

$W_0 > 1$, 螺旋周线向原点收缩。

$W_0 < 1$, 螺旋周线向外旋出。

3.5 线性调频Z变换 (CZT)

CZT计算

(1) 令
$$g(n) = x(n)A^{-n}W^{n^2/2}$$

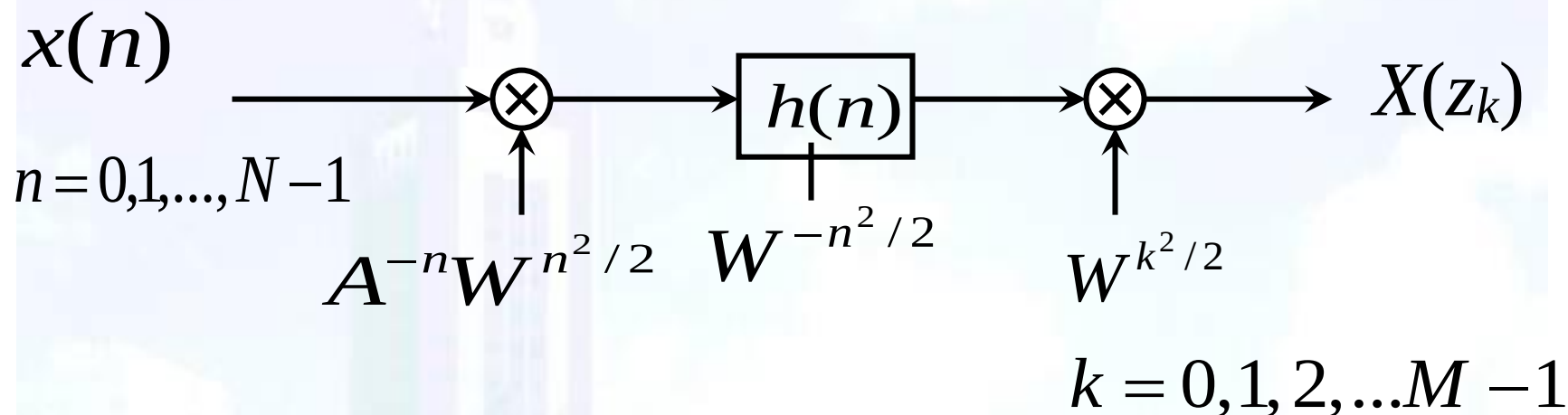
$$h(n) = W^{-n^2/2}$$

则

$$\begin{aligned} X(z_k) \\ = W^{k^2/2} \sum_{n=0}^{N-1} g(n)h(k-n) \quad k = 0, 1, \dots, M-1 \end{aligned}$$

3.5 线性调频Z变换 (CZT)

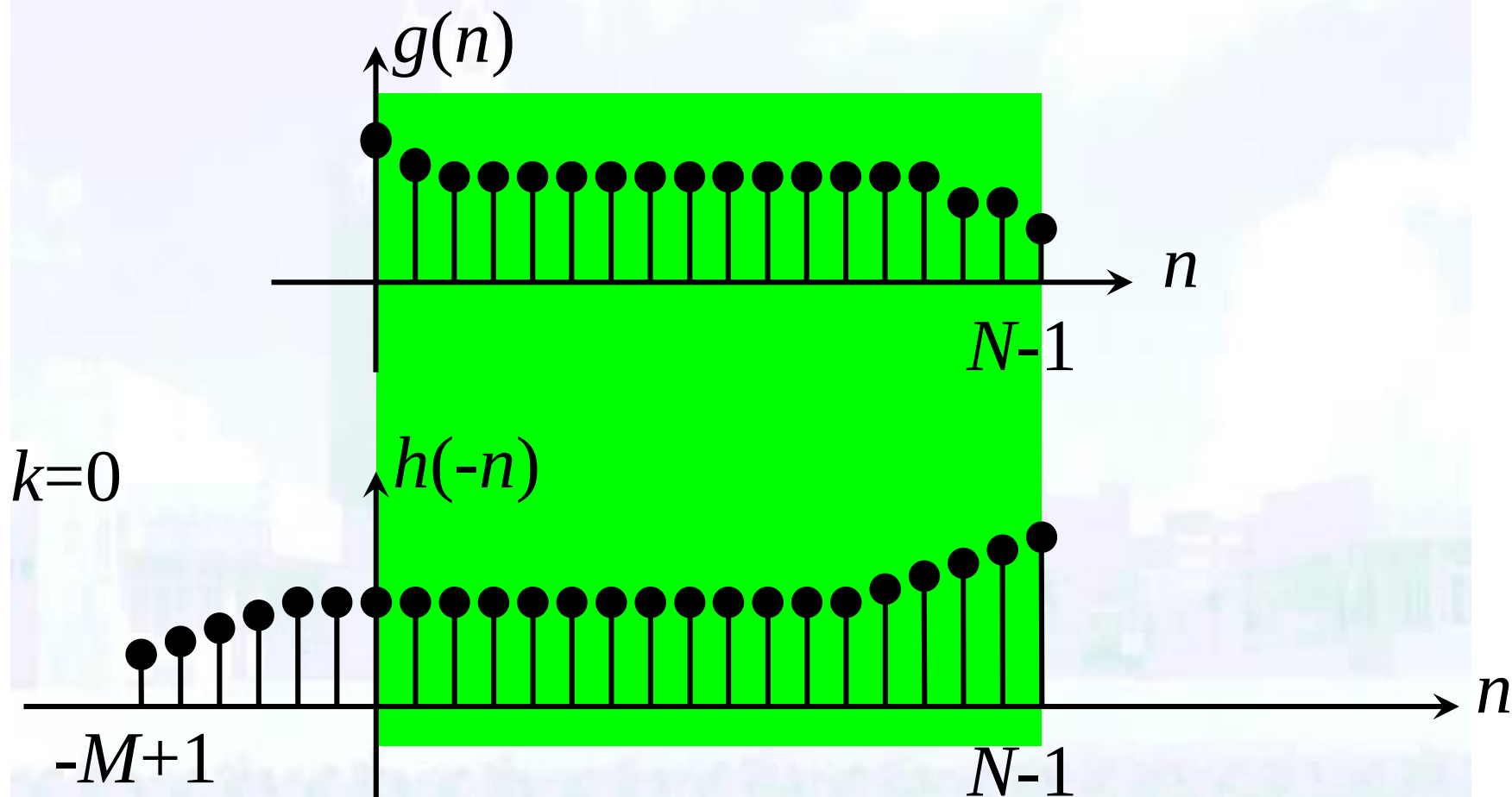
表达为序列 $g(n)$ 通过系统 $h(n)$



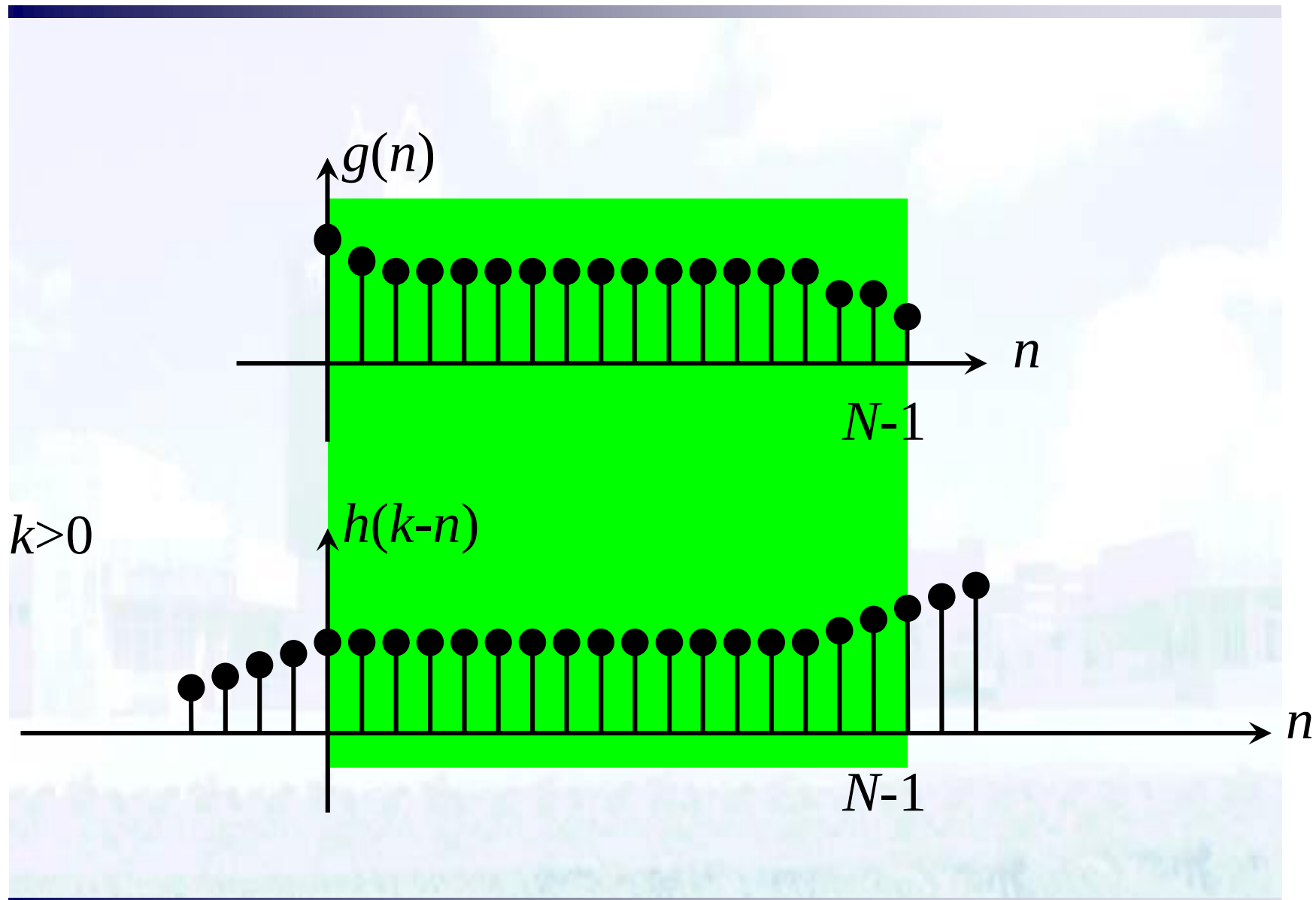
(2)用圆周卷积表示以上线性卷积，设计合适的序列表达，使圆周卷积与线性卷积等价

3.5 线性调频Z变换 (CZT)

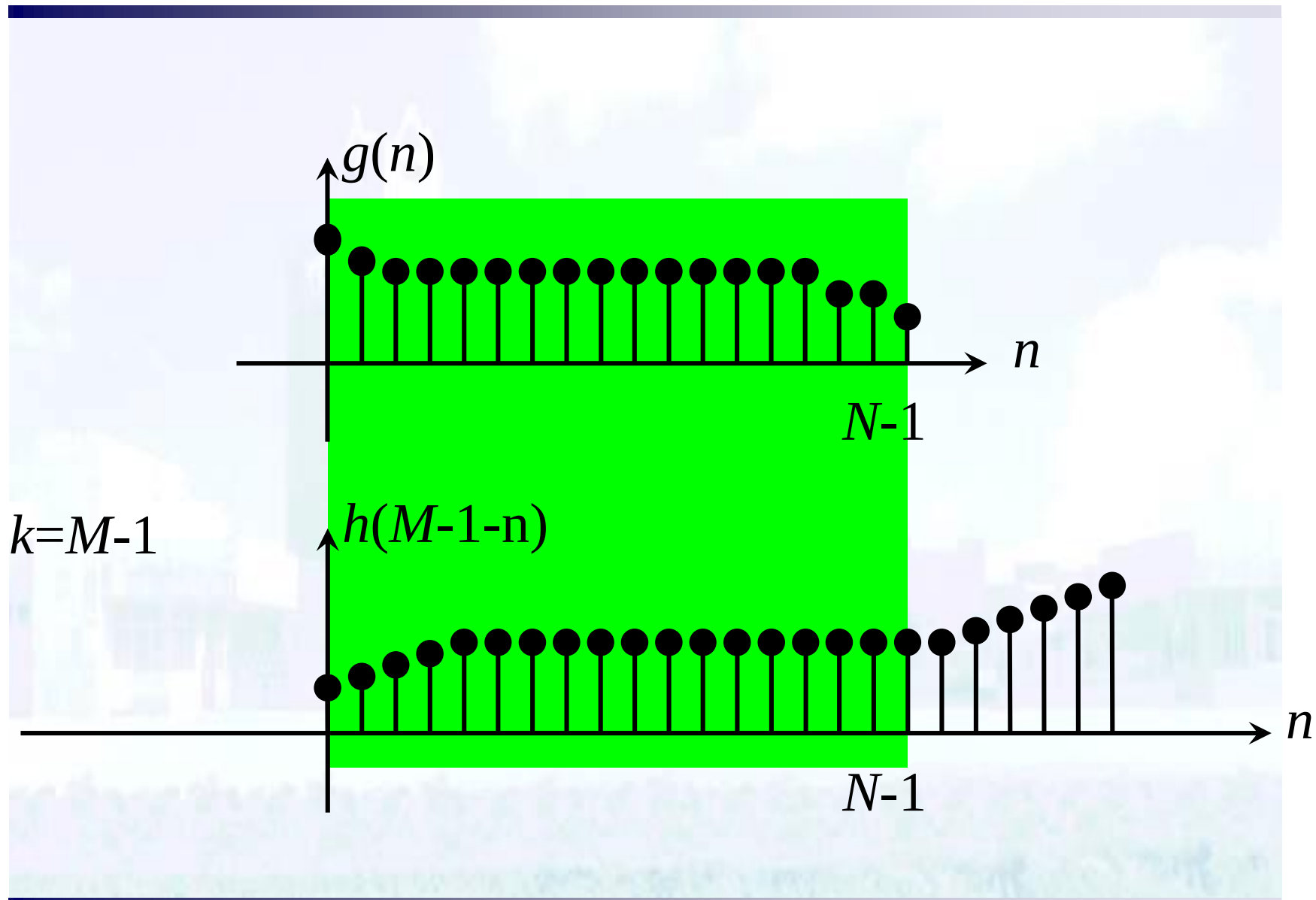
线性卷积过程



3.5 线性调频Z变换 (CZT)

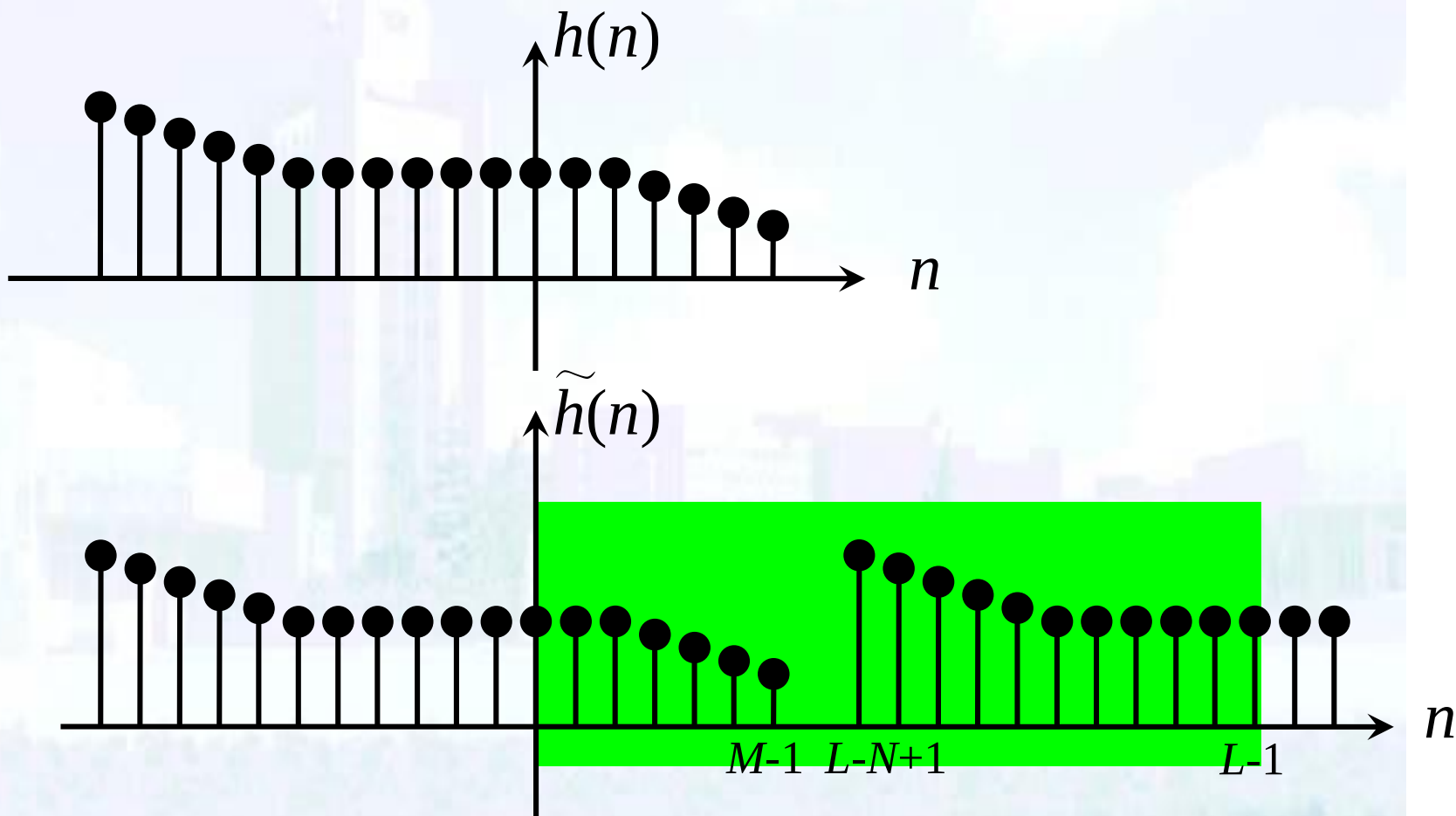


3.5 线性调频Z变换 (CZT)

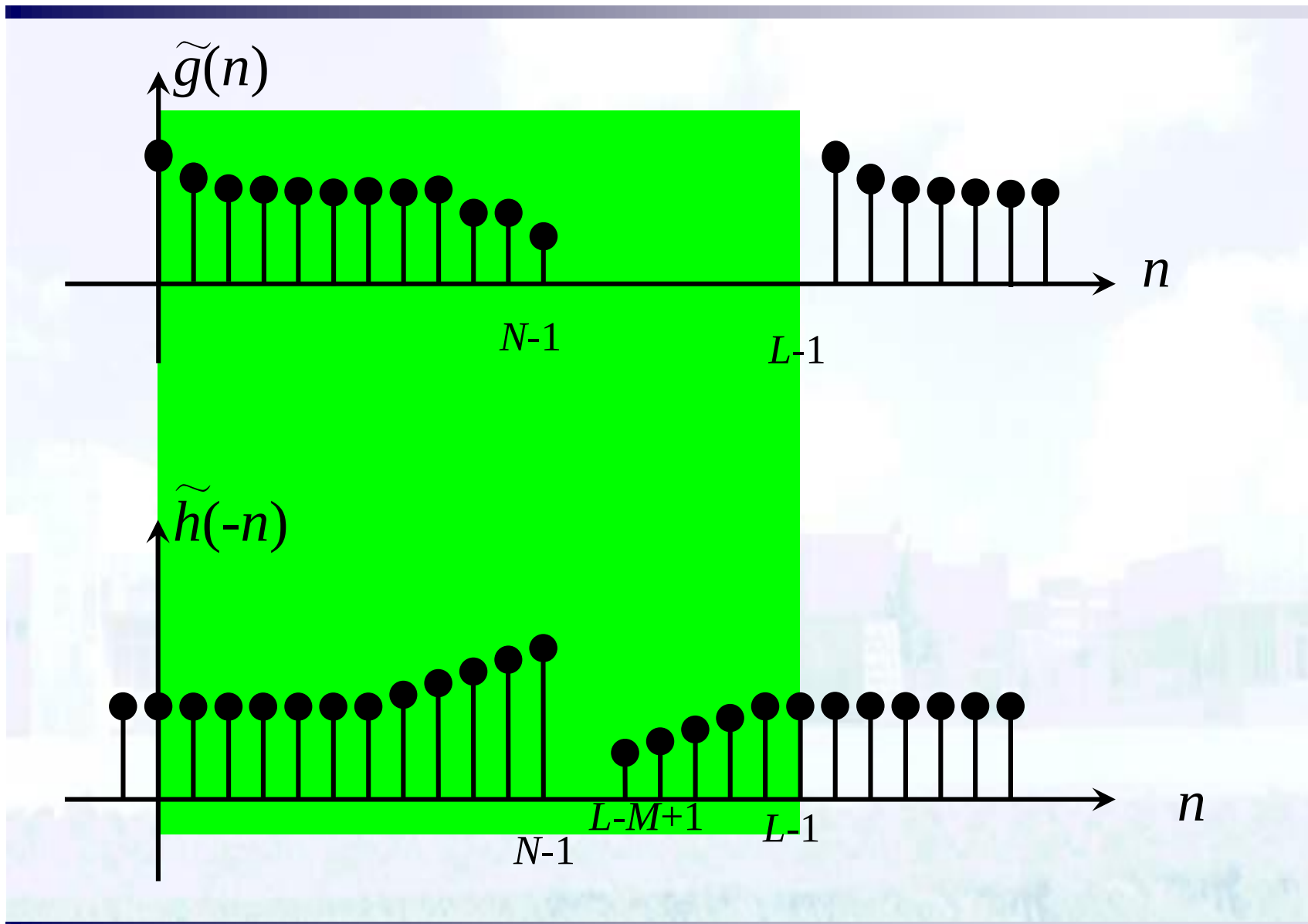


3.5 线性调频Z变换 (CZT)

构造周期序列实现线性卷积



3.5 线性调频Z变换 (CZT)



3.5 线性调频Z变换 (CZT)

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

$$\tilde{g}(n) = \begin{cases} A^{-n} W^{n^2/2} x(n) & n = 0, 1, \dots, N-1 \\ 0 & n = N, \dots, L-1 \end{cases}$$

$$L = 2^r \geq N + M - 1$$

圆周卷积共有 L 个值，前 M 个值($0 \sim M-1$)是所需要的

3.5 线性调频Z变换 (CZT)

计算过程:

乘法次数

$$(1) \tilde{g}(n) = A^{-n} W^{n^2/2} x(n)$$

N

$$(2) \tilde{g}(n) \xrightarrow{FFT} G(k)$$

$(L \log_2 L) / 2$

$$(3) \tilde{h}(n) \xrightarrow{FFT} H(k)$$

$(L \log_2 L) / 2$

$$(4) G(k) \cdot H(k)$$

L

$$(5) IFFT[G(k) \cdot H(k)]$$

$(L \log_2 L) / 2$

$$(6) W^{k^2/2} \cdot IFFT[G(k) \cdot H(k)]$$

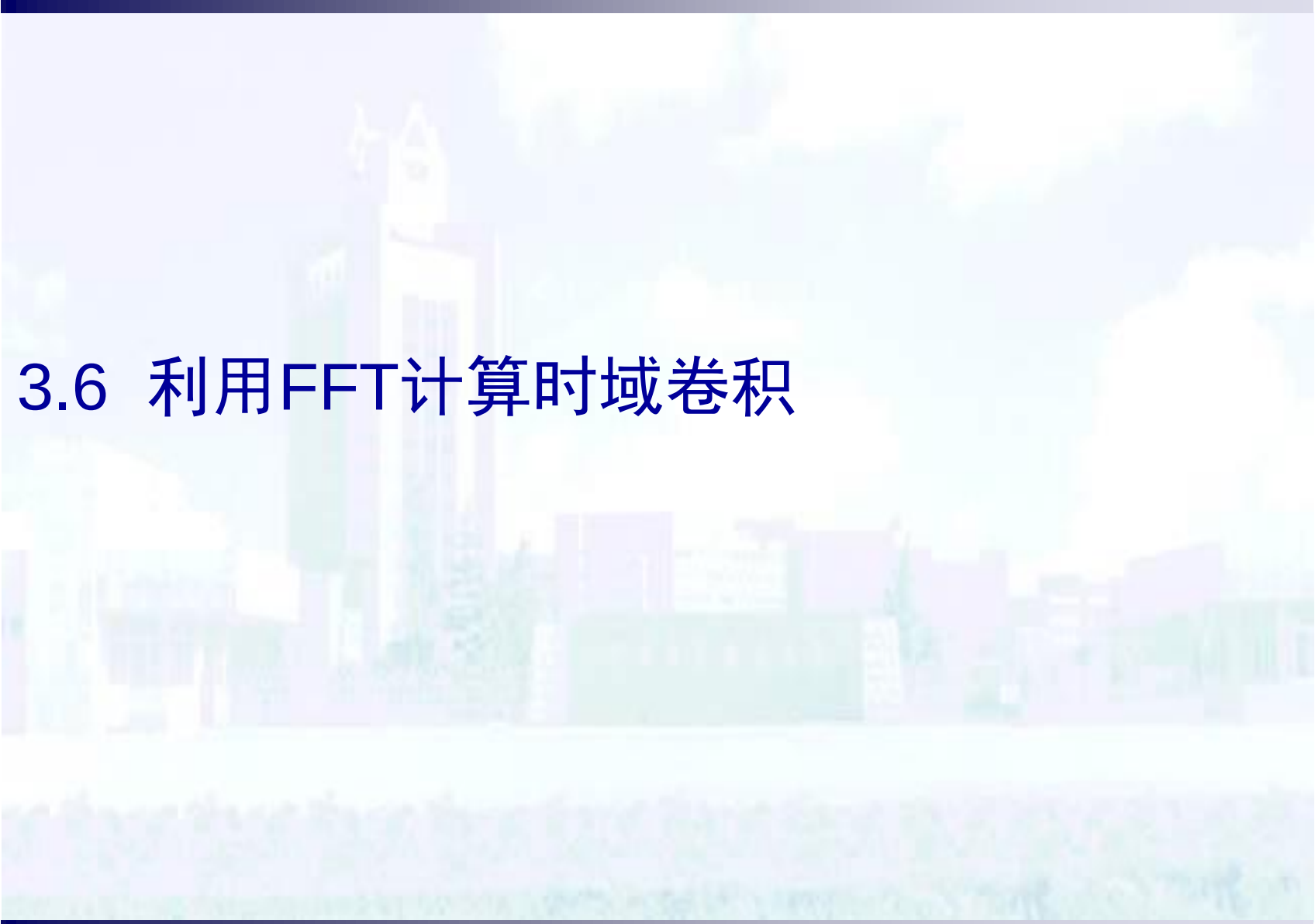
M

3.5 线性调频Z变换 (CZT)

若 $M = N, A=1, W_0=1, \varphi_0 = \frac{2\pi}{N}$

则
$$X(z_k) = \sum_{n=0}^{N-1} x(n) e^{-j \frac{2\pi}{N} nk}$$

CZT退化成为 $x(n)$ 的DFT。也可以采用上述方法计算，即所谓的素数DFT快速计算。



3.6 利用FFT计算时域卷积

线性卷积的频域计算

序列 $x(n)$ 长度为 N , $h(n)$ 长度为 M 。

线性卷积

$$\begin{aligned} y(n) &= \sum_{k=-\infty}^{\infty} x(k)h(n-k) \\ &= \sum_{k=0}^{N-1} x(k)h(n-k) \end{aligned}$$

3.6 利用FFT计算时域卷积

$x(k)$ 与 $h(n-k)$ 的有效取值范围:

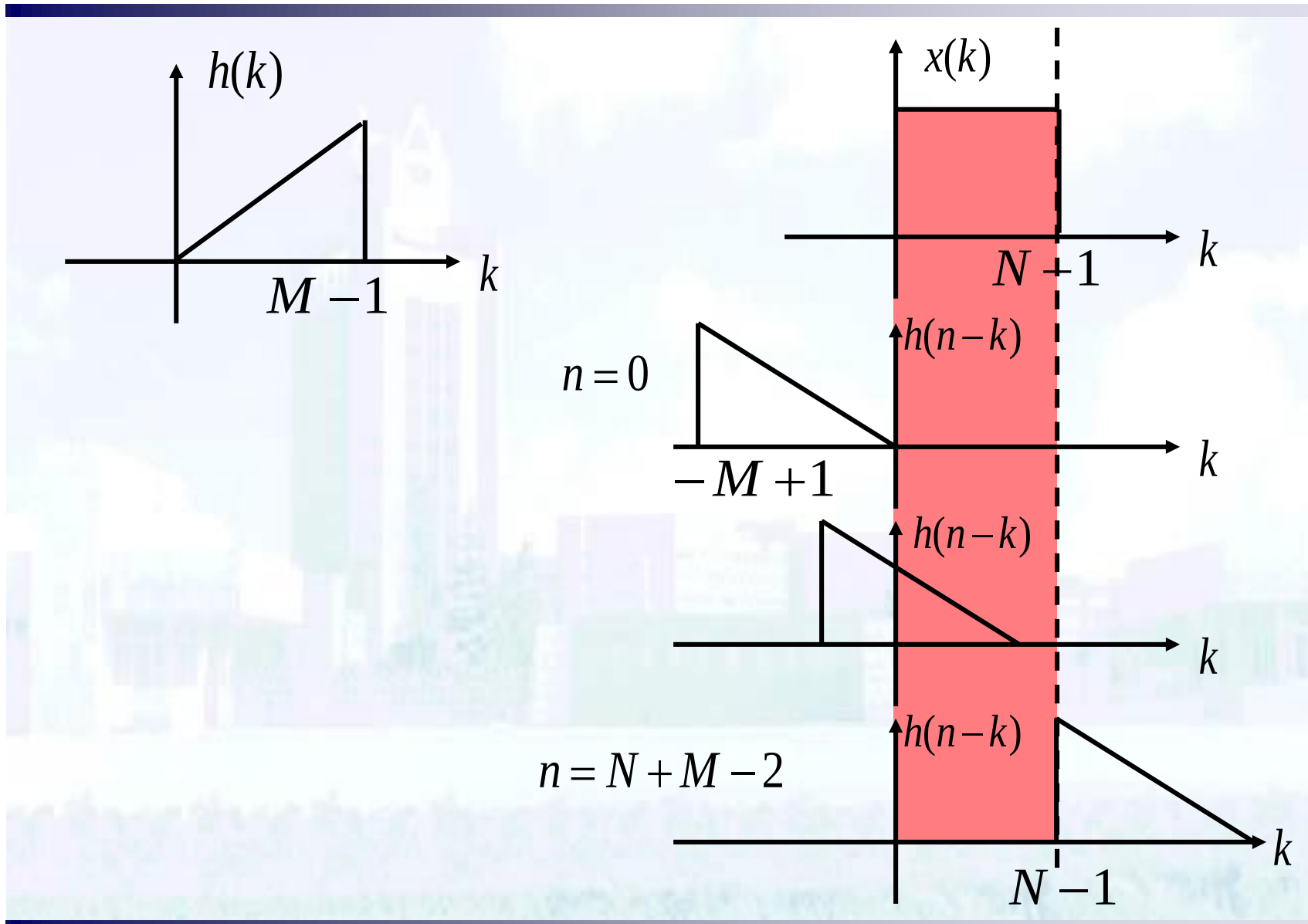
$$0 \leq k \leq N - 1$$

$$0 \leq n - k \leq M - 1$$

$$\therefore 0 \leq n \leq M + N - 2$$

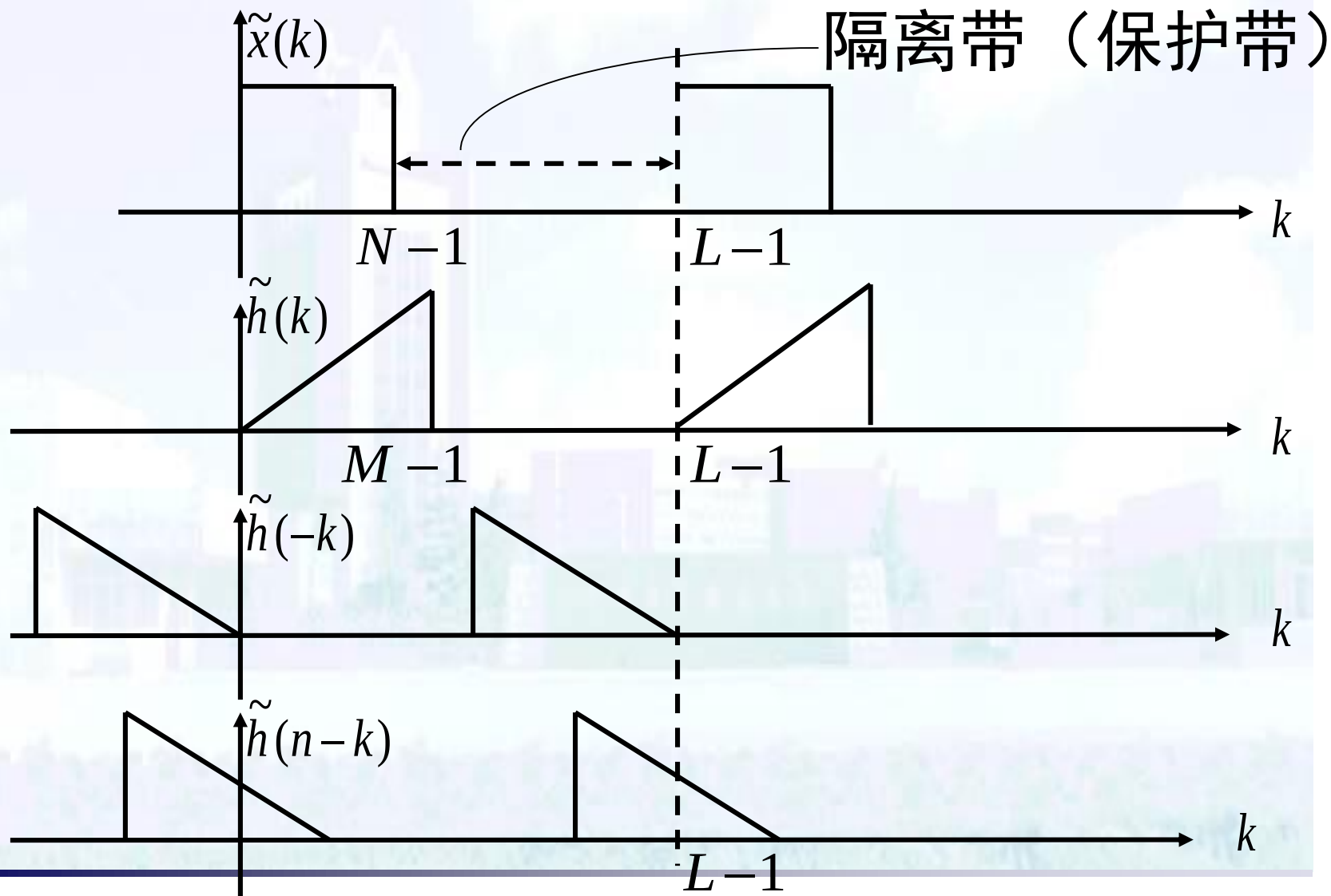
$y(n)$ 有 $M+N-1$ 个有效值

3.6 利用FFT计算时域卷积

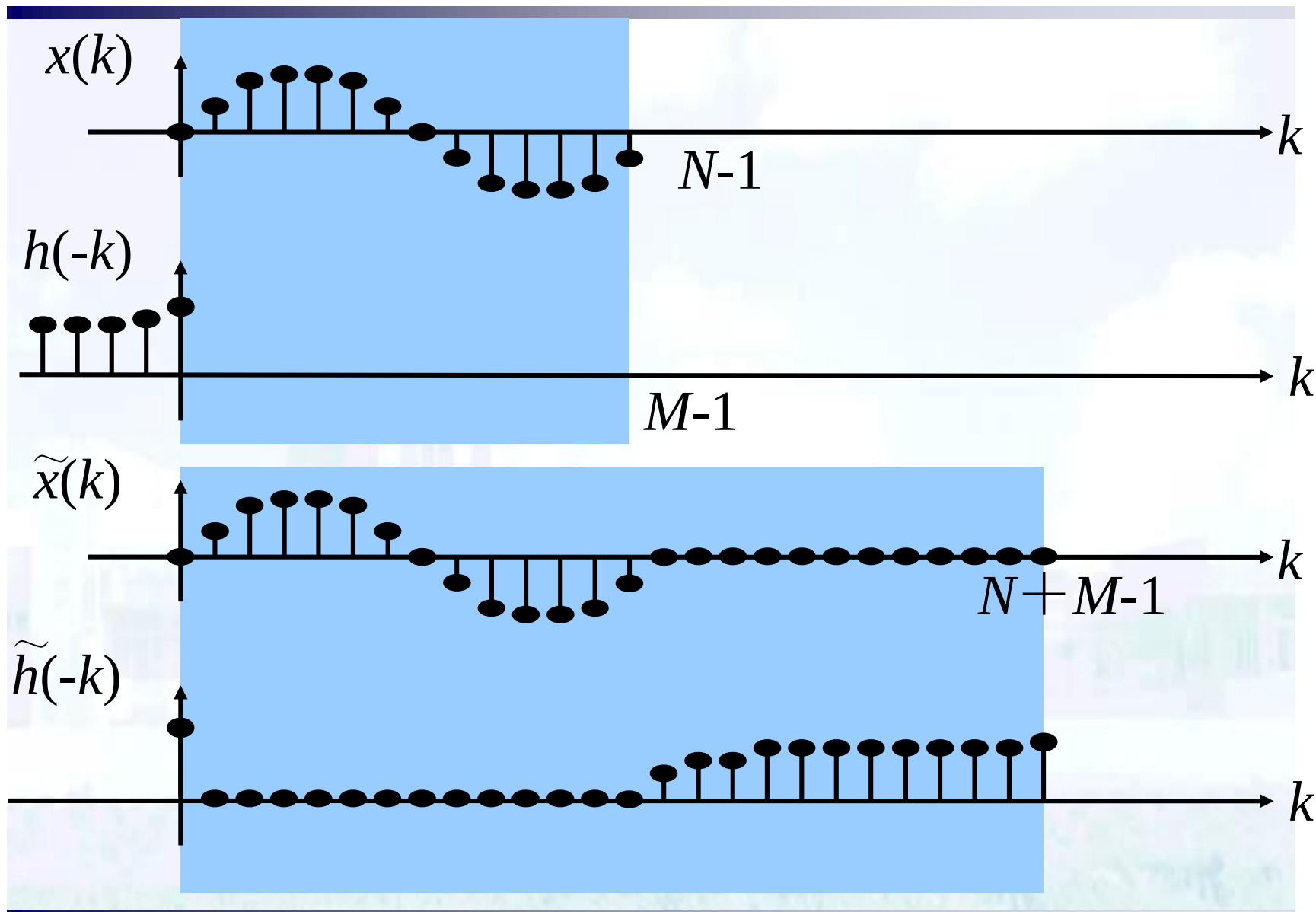


3.6 利用FFT计算时域卷积

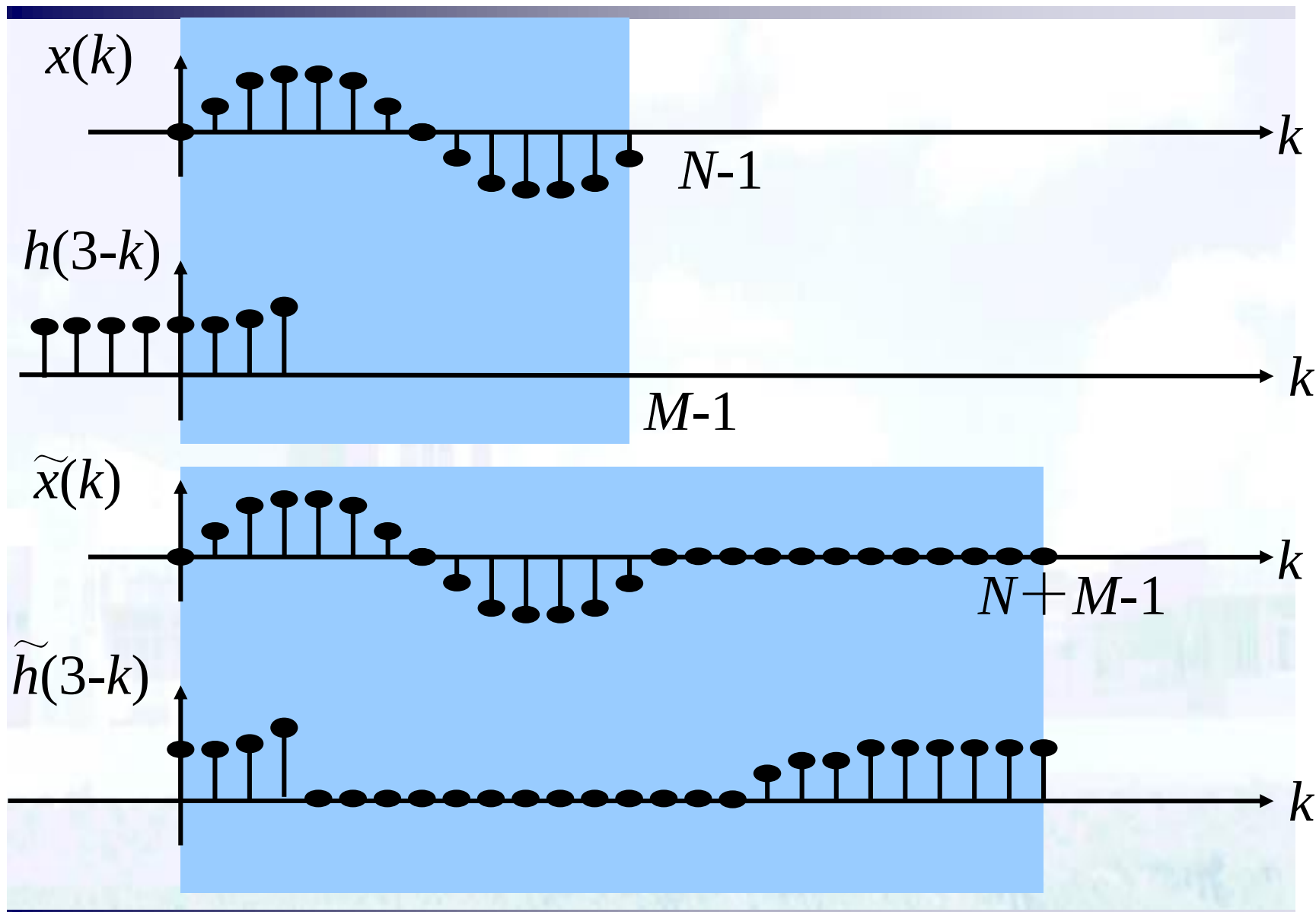
用圆周卷积表达线性卷积：构造周期序列



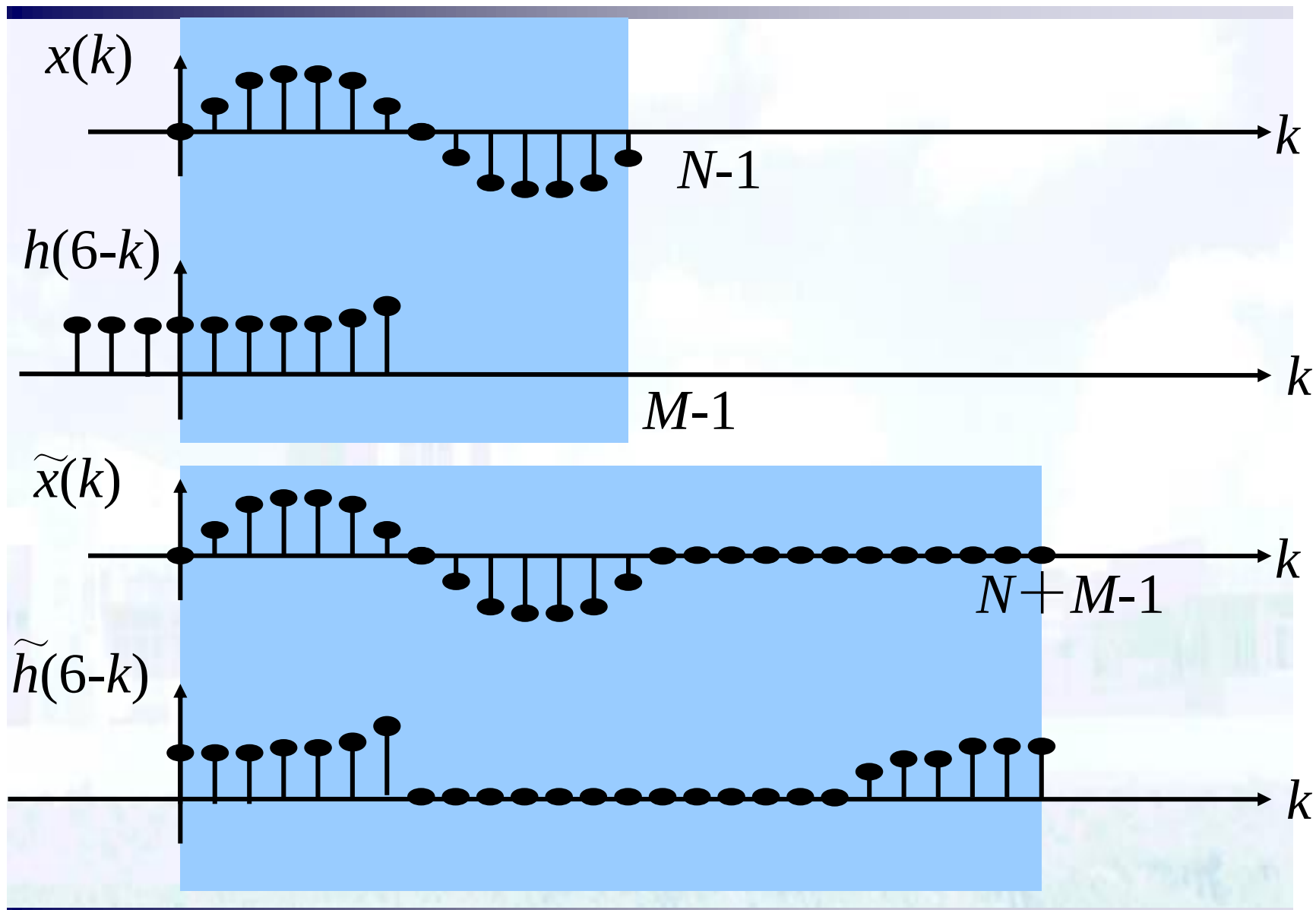
3.6 利用FFT计算时域卷积



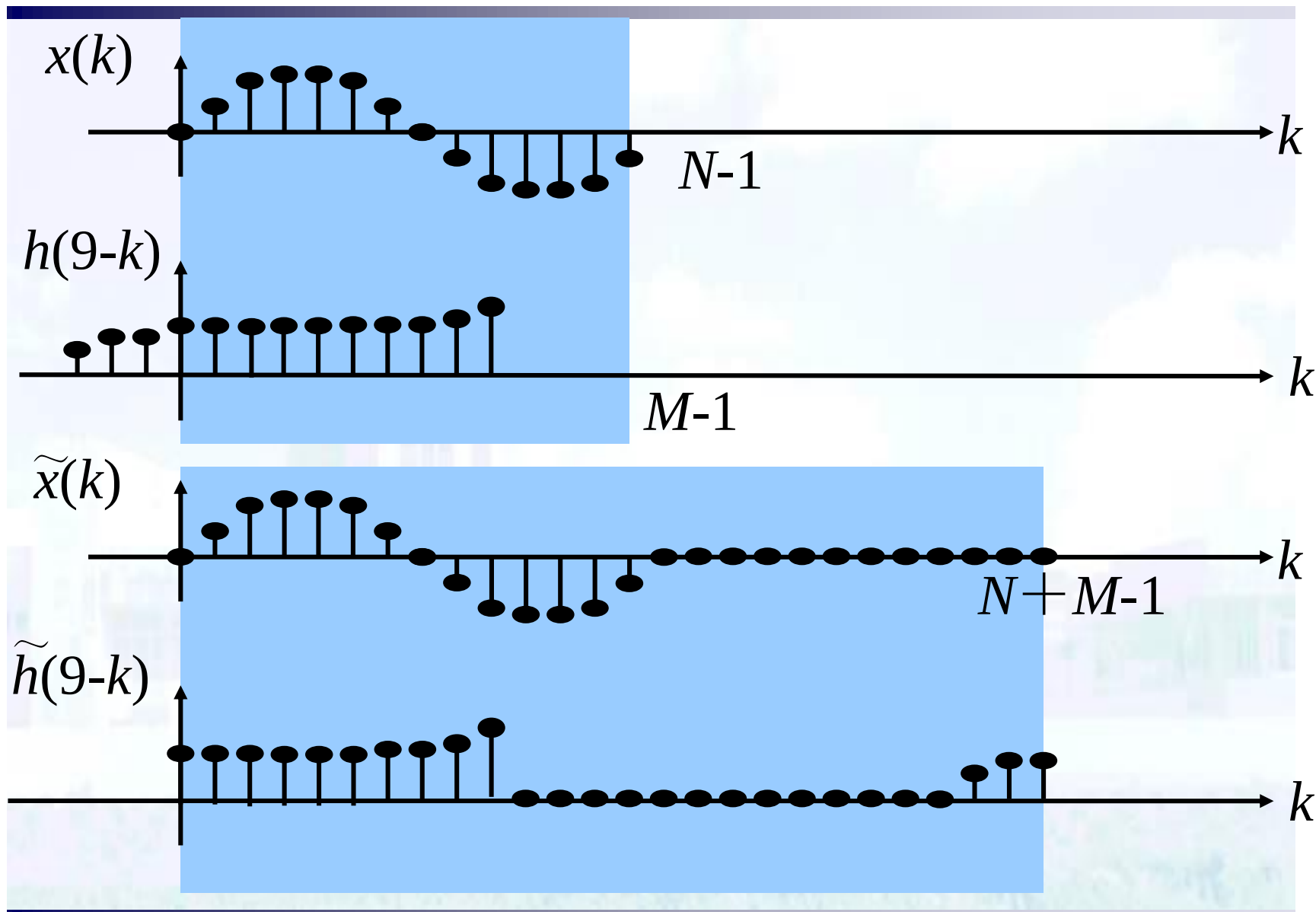
3.6 利用FFT计算时域卷积



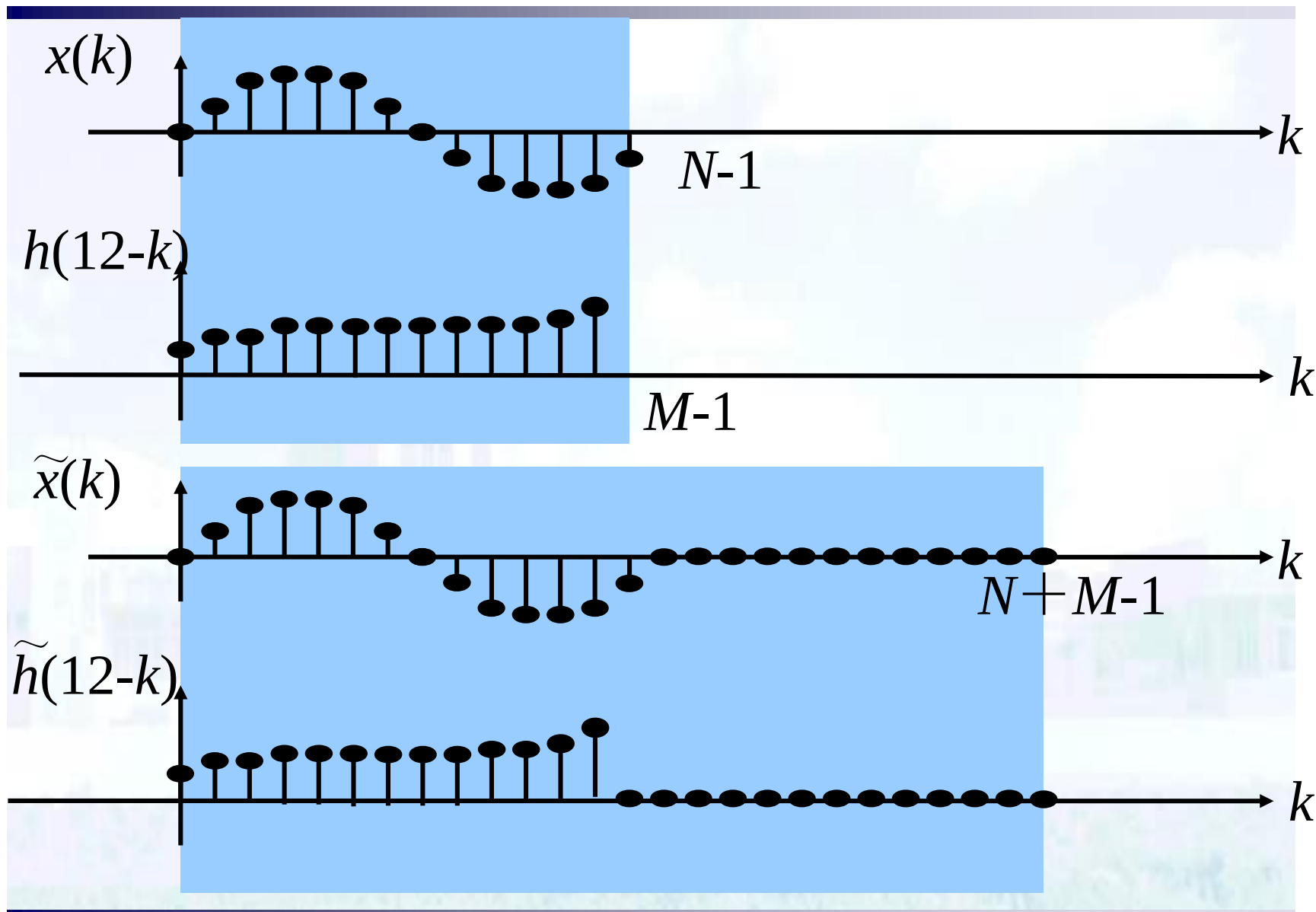
3.6 利用FFT计算时域卷积



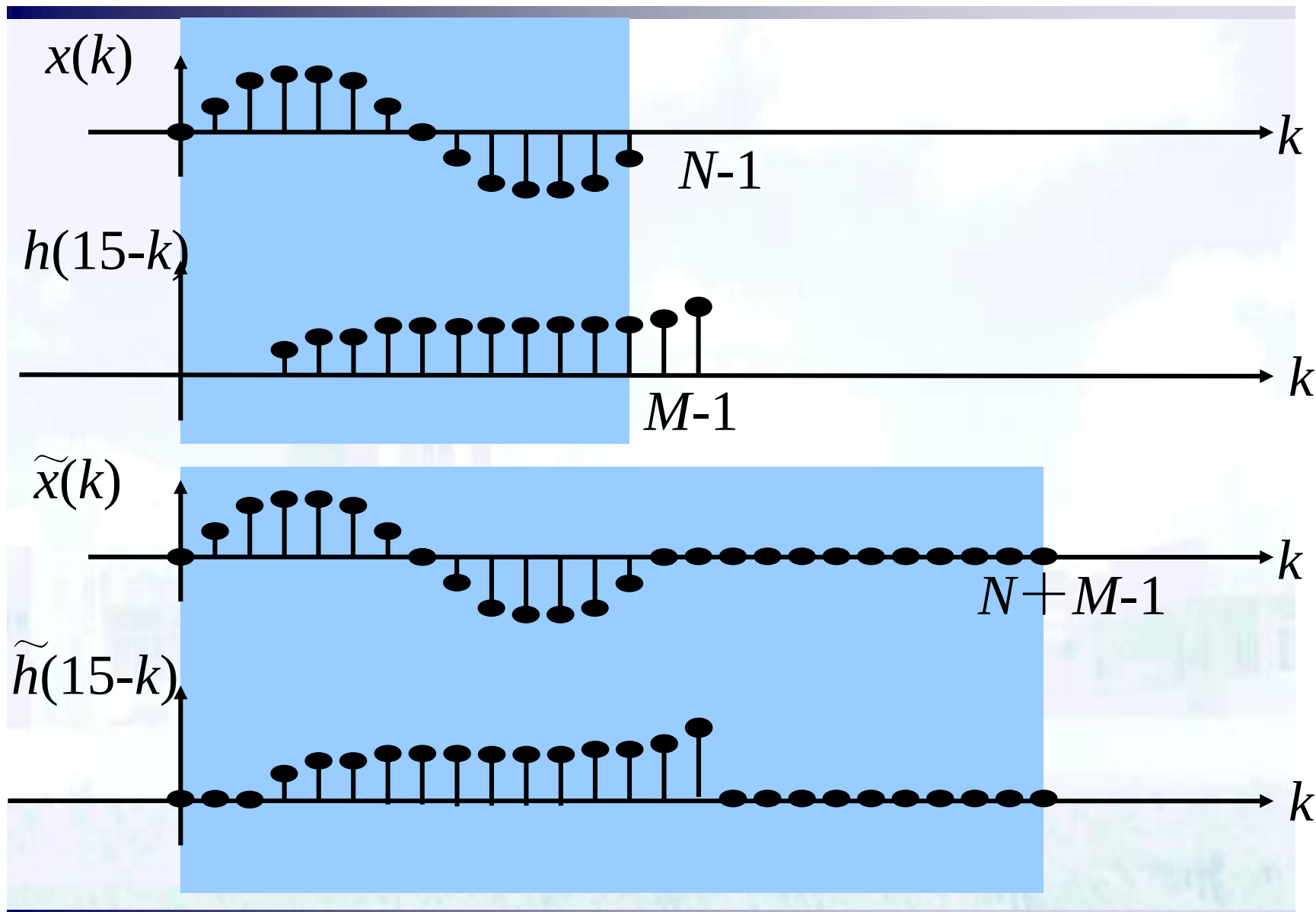
3.6 利用FFT计算时域卷积



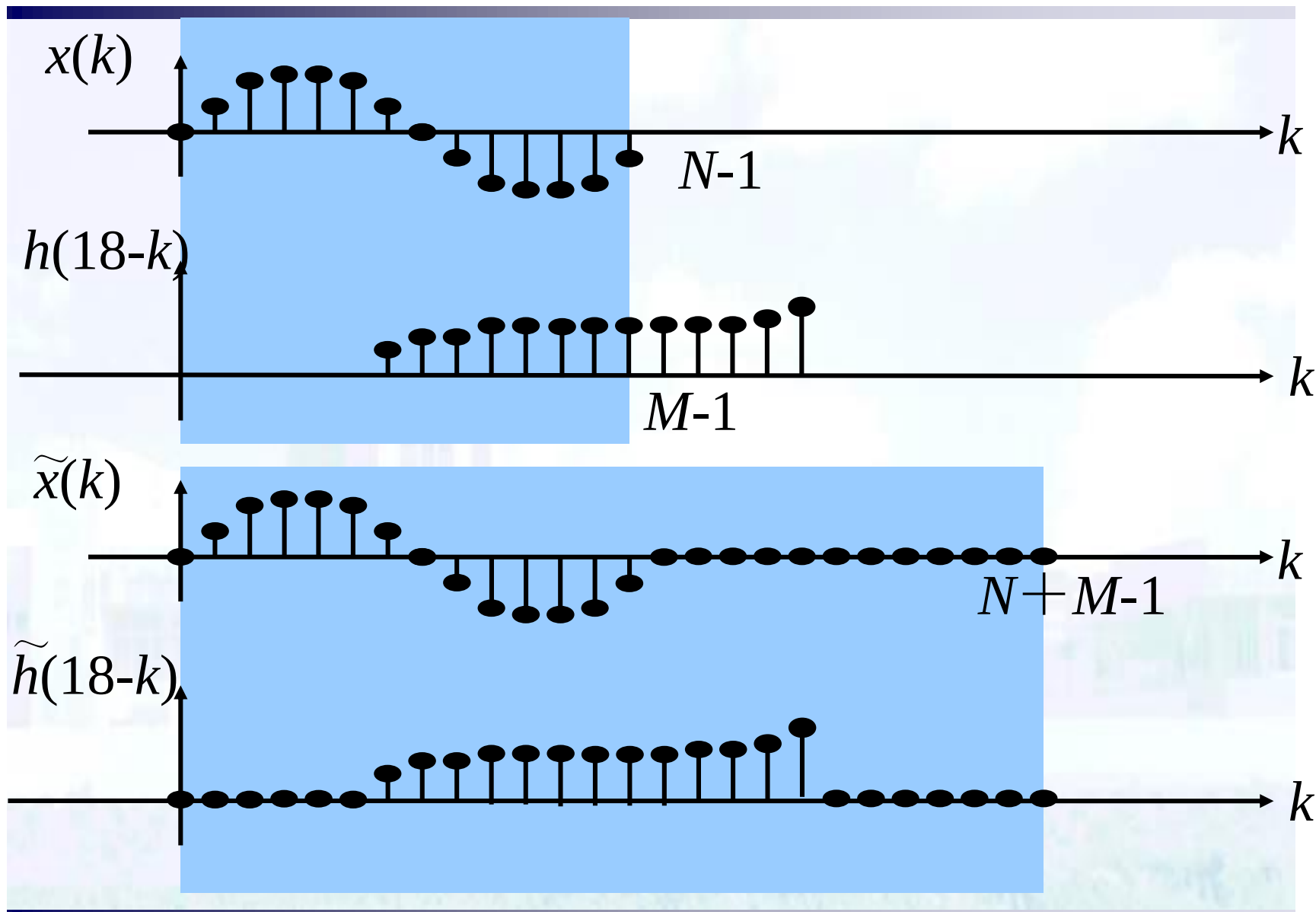
3.6 利用FFT计算时域卷积



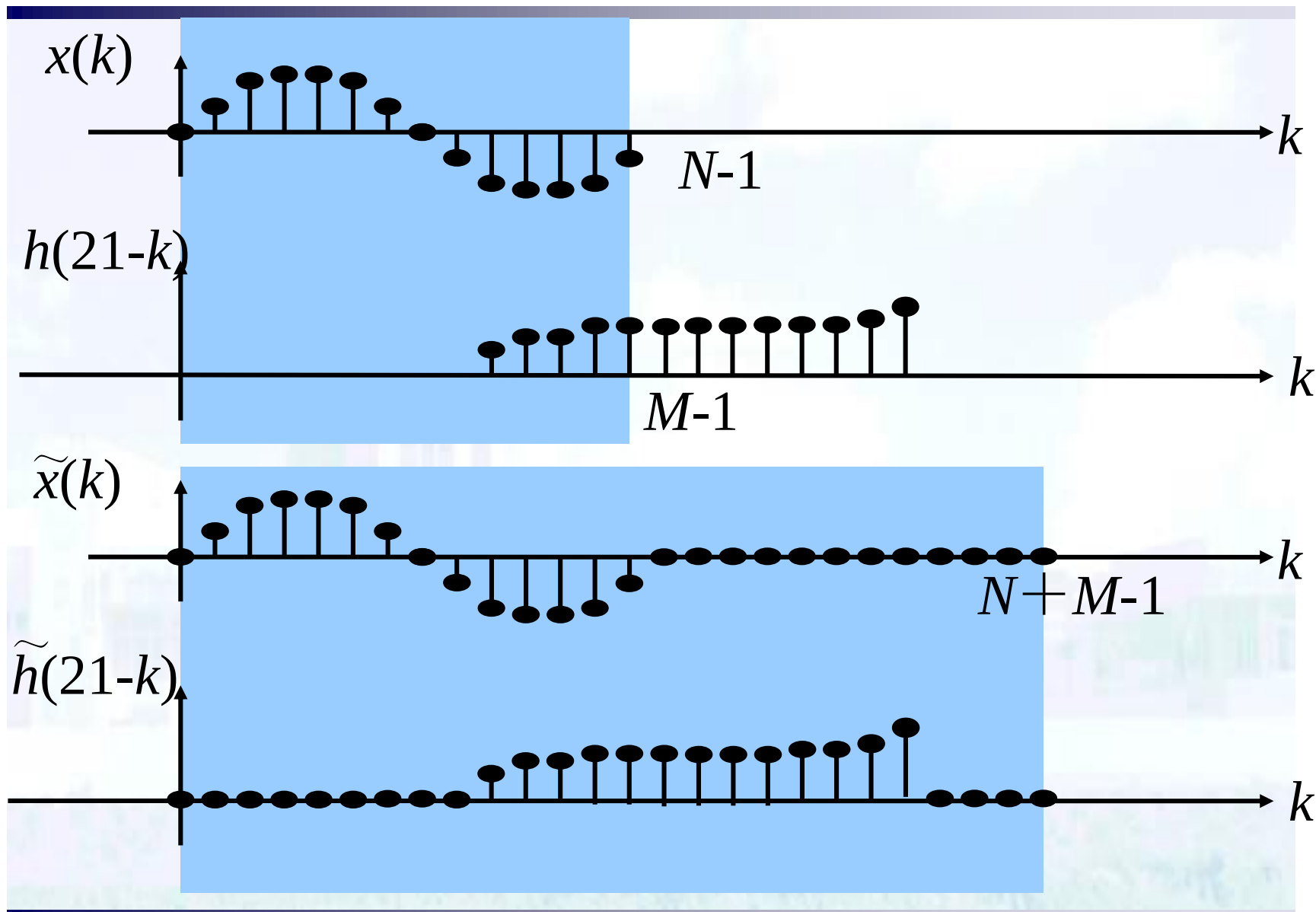
3.6 利用FFT计算时域卷积



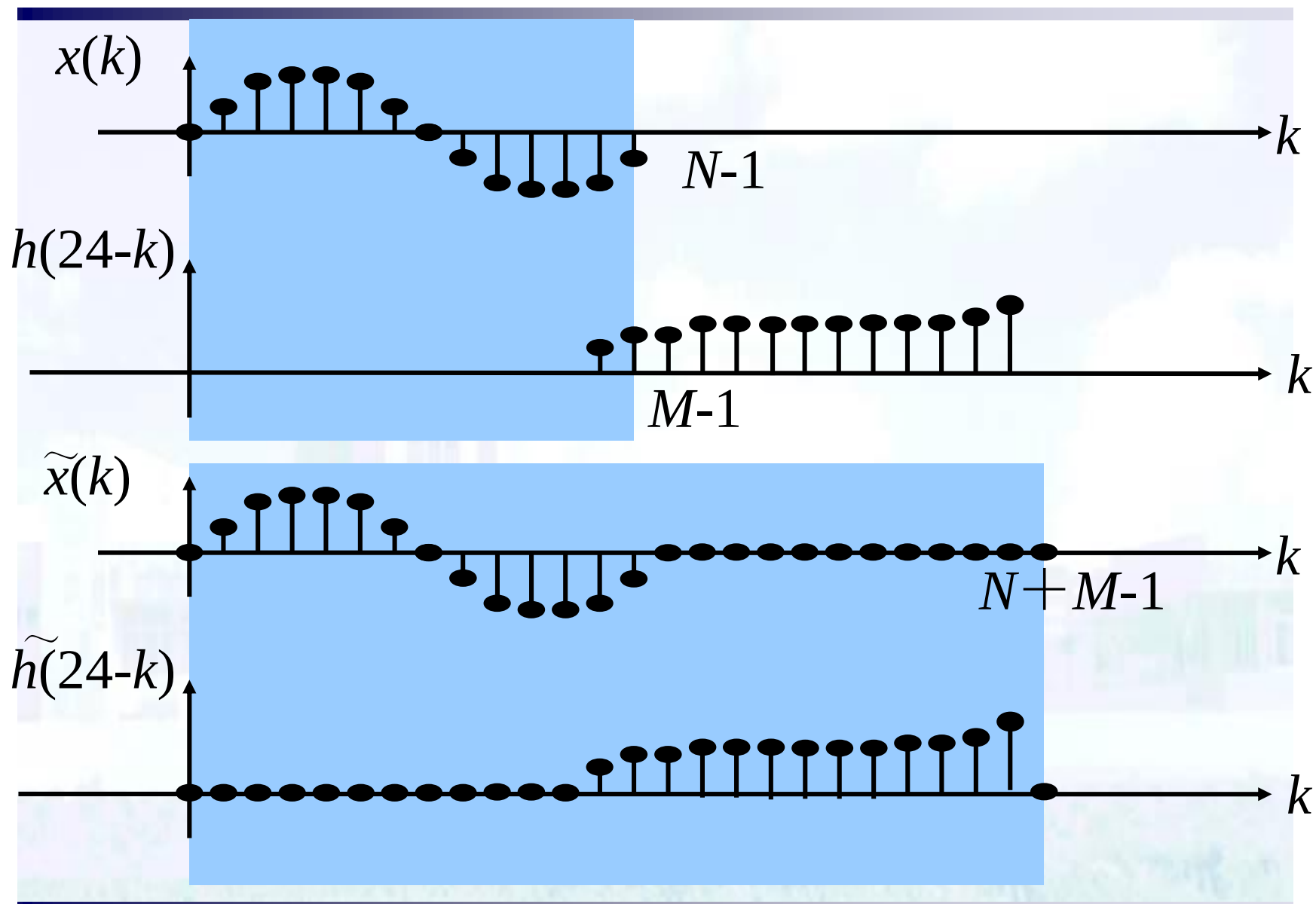
3.6 利用FFT计算时域卷积



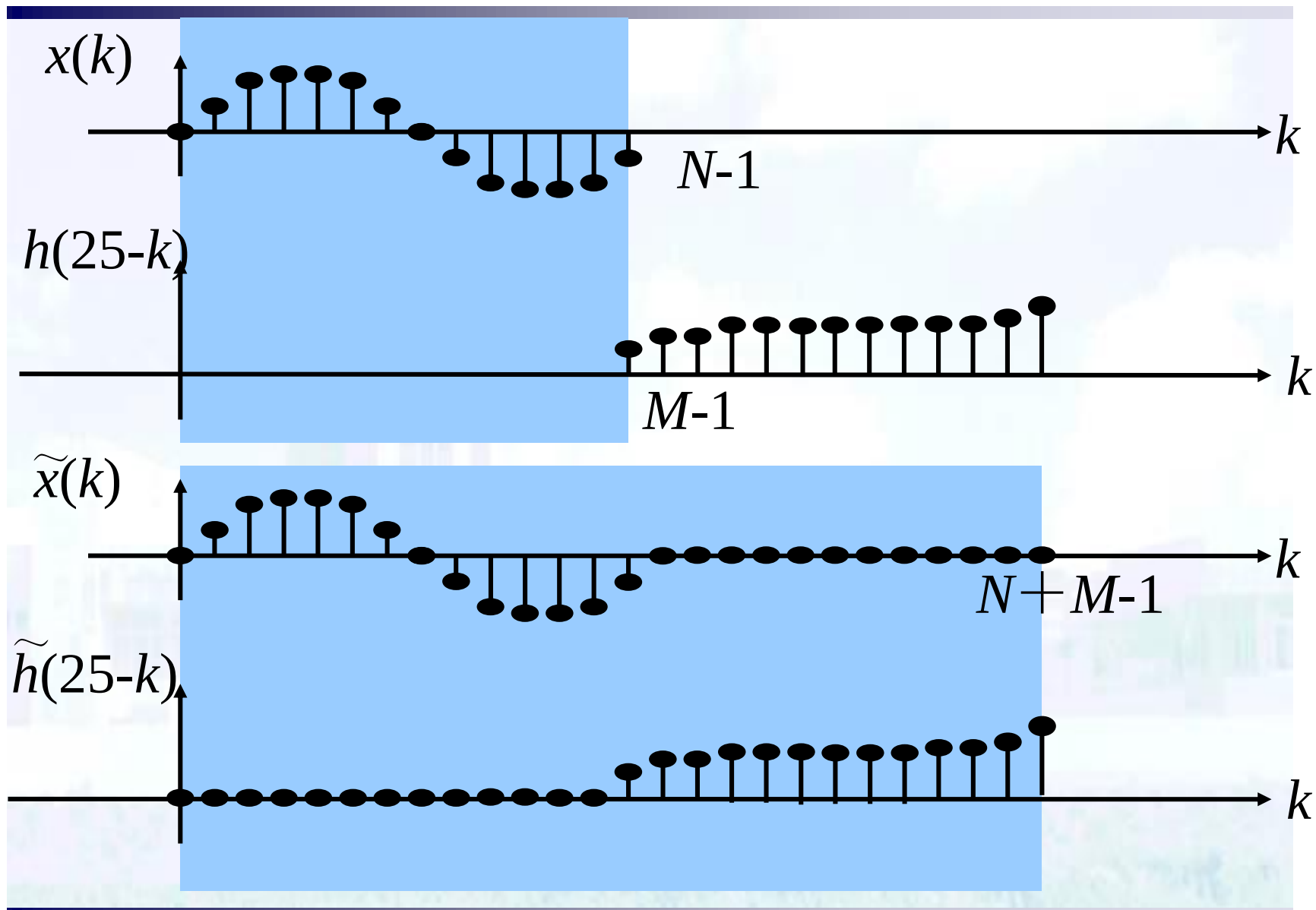
3.6 利用FFT计算时域卷积



3.6 利用FFT计算时域卷积



3.6 利用FFT计算时域卷积



3.6 利用FFT计算时域卷积

周期序列长度 L 的选取:

$$x_L(n) = \begin{cases} x(n) & 0 \leq n \leq N-1 \\ 0 & N \leq n \leq L-1 \end{cases}$$

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

$$L \geq N + M - 1$$

利用FFT, 要求 $L = 2^v \geq N + M - 1$

3.6 利用FFT计算时域卷积

圆周卷积结果,

$$\begin{aligned} y_L(n) &= x_L(n) \otimes h_L(n) \\ &= IDFT[X_L(k) \cdot H_L(k)] \end{aligned}$$

$$X_L(k) = DFT[x_L(n)]$$

$$H_L(k) = DFT[h_L(n)]$$

3.6 利用FFT计算时域卷积

当 $N \gg M$ 时，常把 N 分为若干段进行卷积：

- (1) 较快地得到输出（低延迟）；
- (2) 减少运算量；
- (3) 需要较少的存储单元。

3.6 利用FFT计算时域卷积

重叠相加法

将 $x(n)$ 分段，每段长 L 。确定 L 的原则：

(1) 与 M 大小相当

(2) $L+M-1=2^v$

$$x_k(n) = \begin{cases} x(n) & kL \leq n \leq (k+1)L-1 \\ 0 & \text{其他} \end{cases}$$

$$x(n) = \sum_{k=0}^K x_k(n)$$

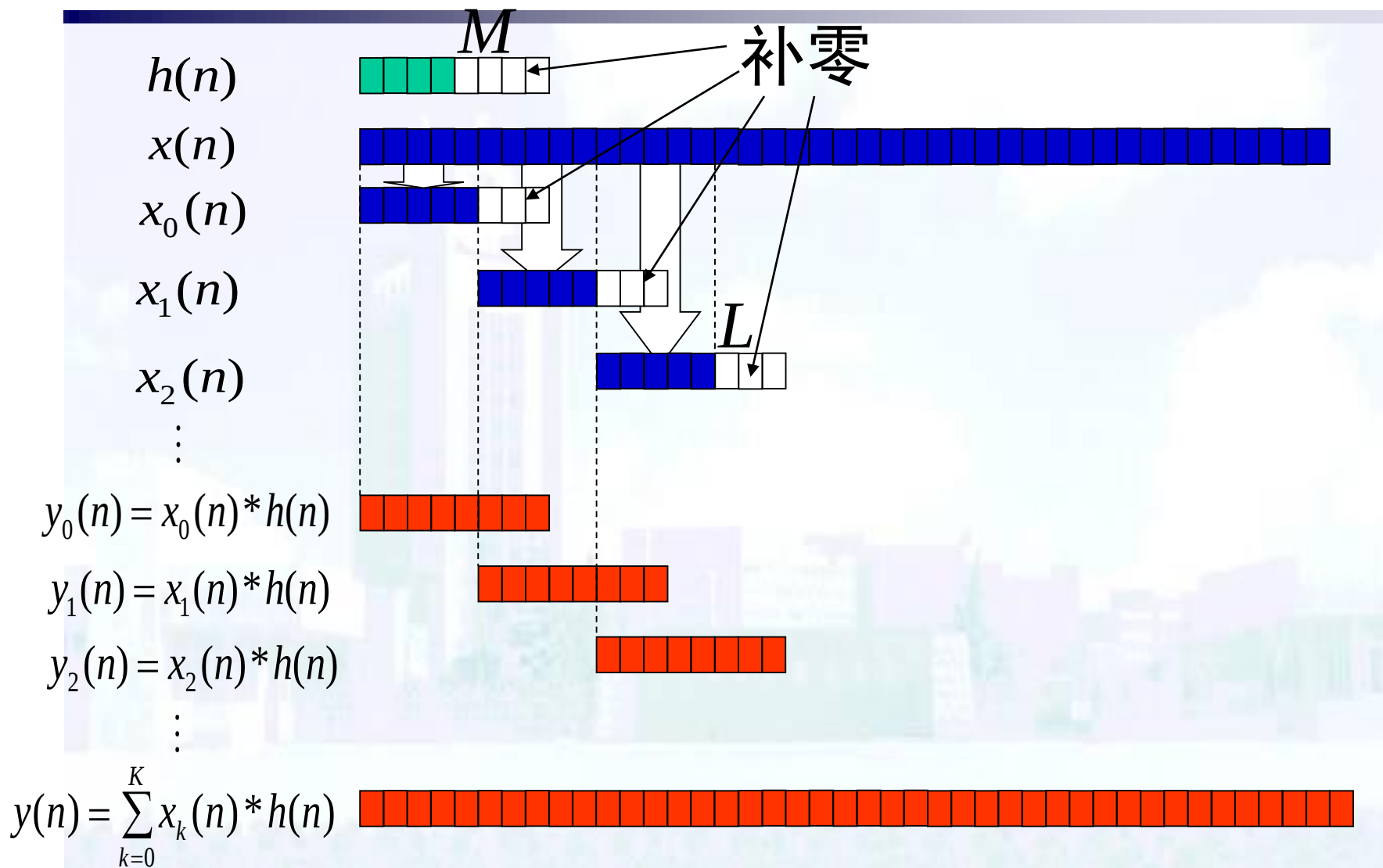
3.6 利用FFT计算时域卷积

$$y(n) = x(n) * h(n) = \sum_{k=0}^K x_k(n) * h(n)$$

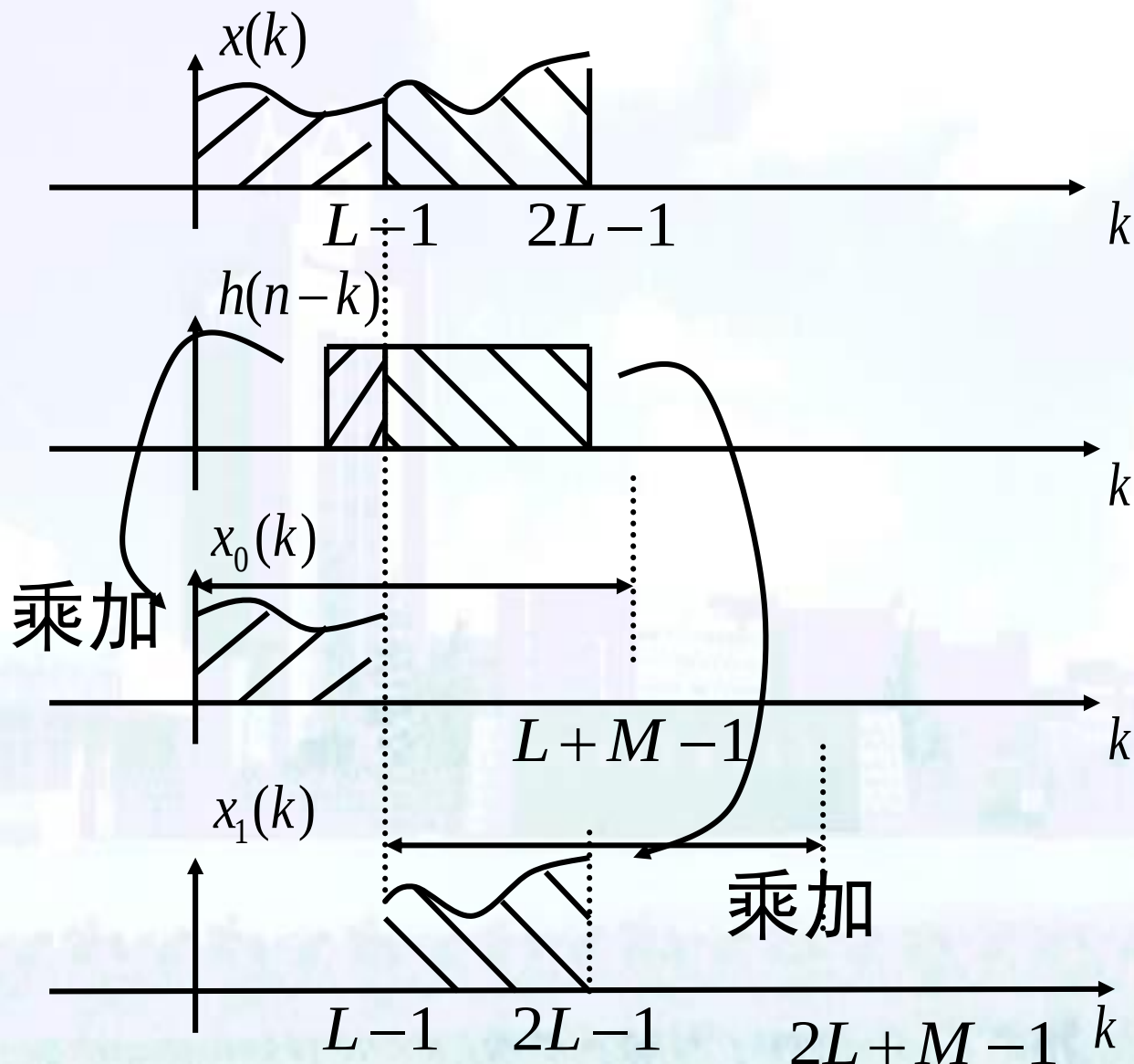
计算 $x_k(n) * h(n)$ 时采用补零的圆周卷积,

将 $x_k(n)$ 和 $h(n)$ 变成长度为 $L+M-1$ 的序列。

3.6 利用FFT计算时域卷积

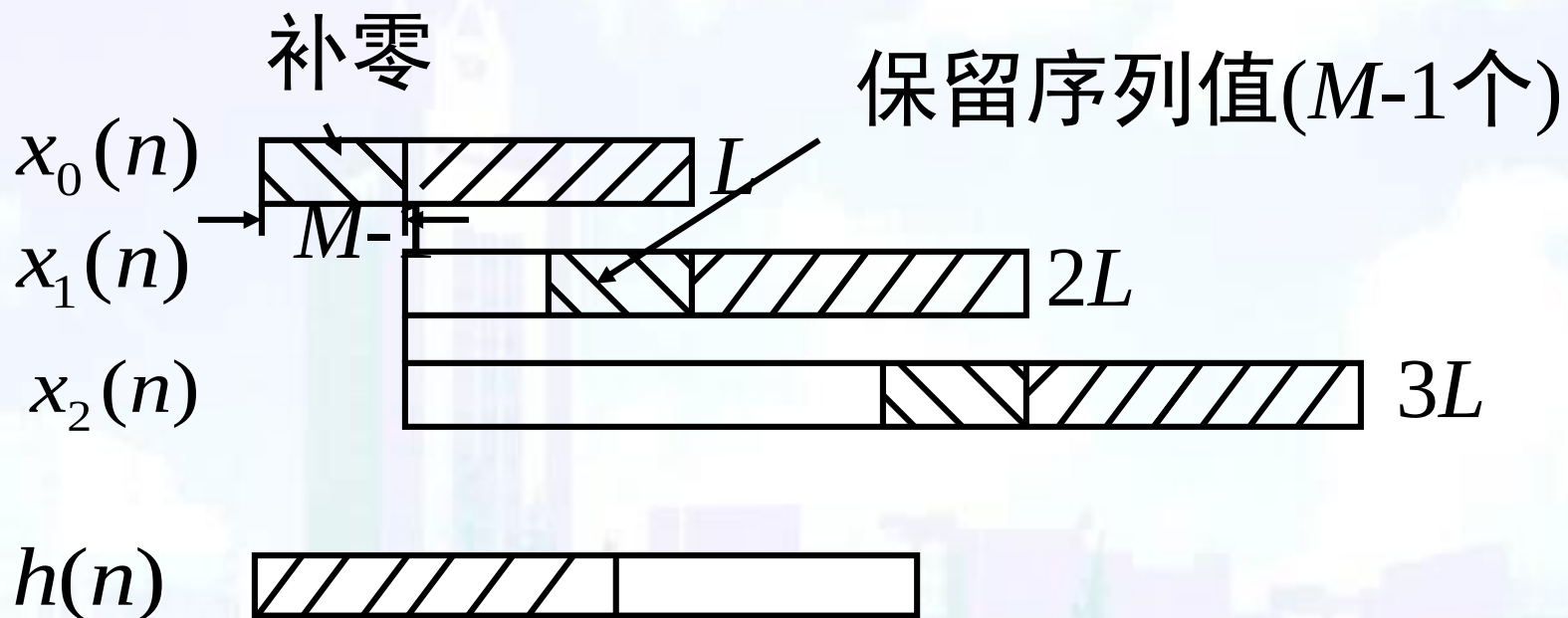


3.6 利用FFT计算时域卷积



3.6 利用FFT计算时域卷积

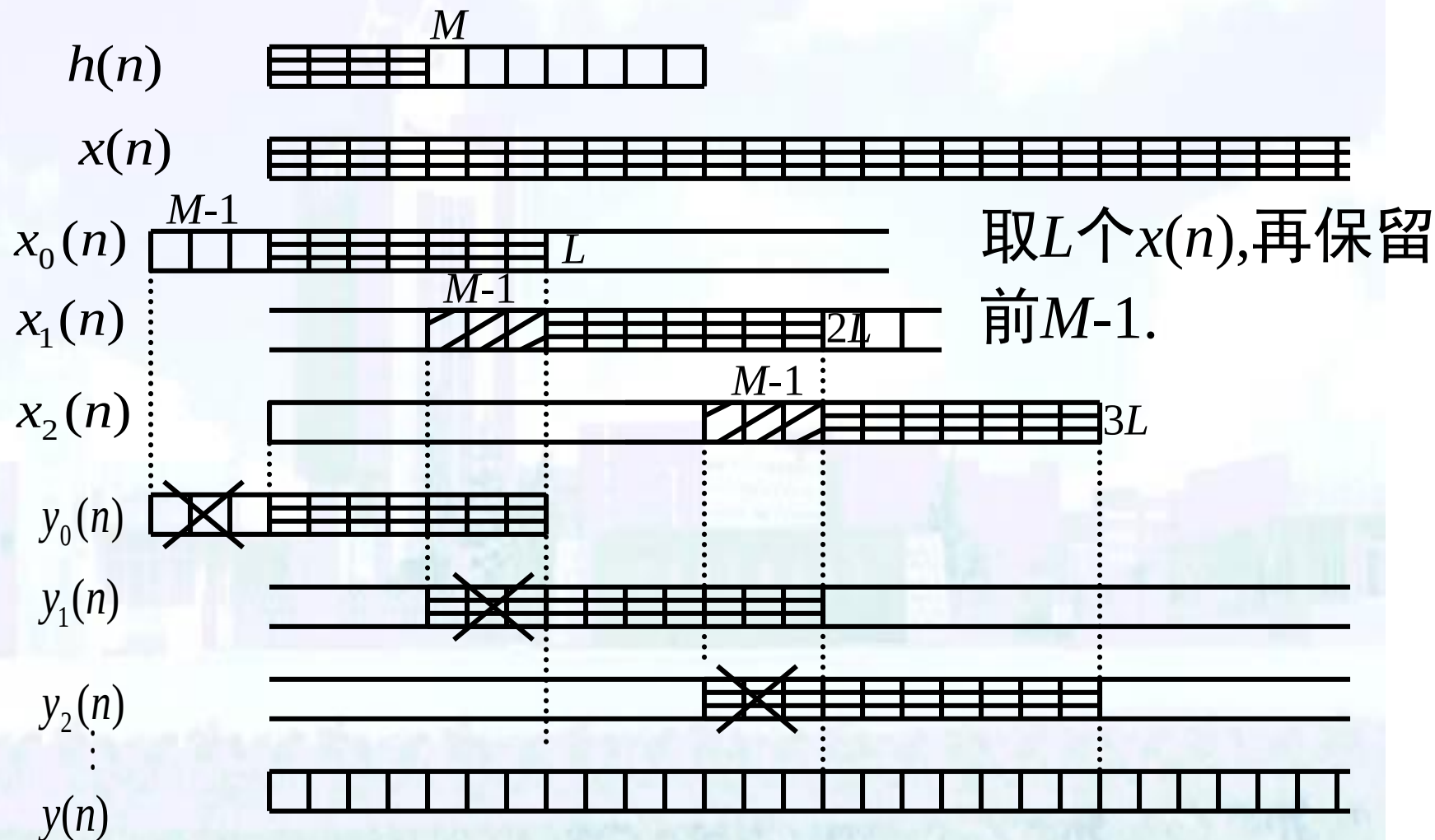
重叠保留法



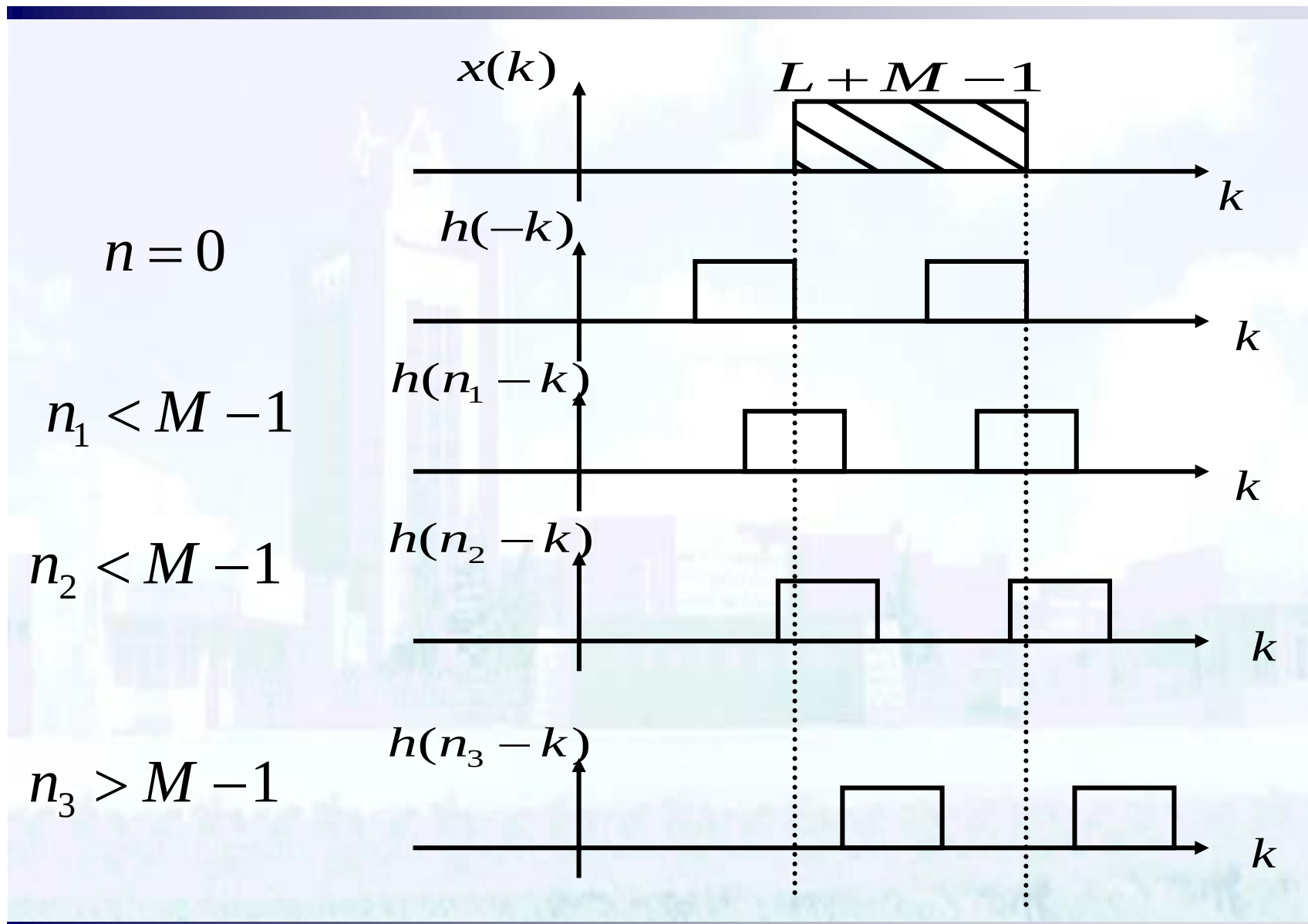
丢弃每段卷积结果的前 $M-1$ 个值

3.6 利用FFT计算时域卷积

运算流程:



3.6 利用FFT计算时域卷积



3.6 利用FFT计算时域卷积

线性相关的频域计算

实序列 $x_1(n)$, 列长 N_1
 $x_2(n)$, 列长 N_2

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

3.6 利用FFT计算时域卷积

$x_1(k)$ 与 $x_2(n+k)$ 均不为零的范围

$$0 \leq k \leq N_1 - 1$$

$$0 \leq k + n \leq N_2 - 1$$

$$-N_1 + 1 \leq n \leq N_2 - 1$$

$\Rightarrow y(n)$ 有 $N_1 + N_2 - 1$ 个有效值

用圆周相关计算，为避免周期项引入的混叠，
对 $x_1(n)$ 和 $x_2(n)$ 作补零处理

3.6 利用FFT计算时域卷积

$$N = 2^v \geq N_1 + N_2 - 1$$

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

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

3.6 利用FFT计算时域卷积

$$\tilde{x}_1(n) \xrightarrow{FFT} X_1(k)$$

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

$$Y(k) = X_1^*(k) X_2(k) \quad k = 0, 1, \dots, N-1$$

$$Y(k) \xrightarrow{FFT} y(n)$$

取前 $N_1 + N_2 - 1$ 个值。



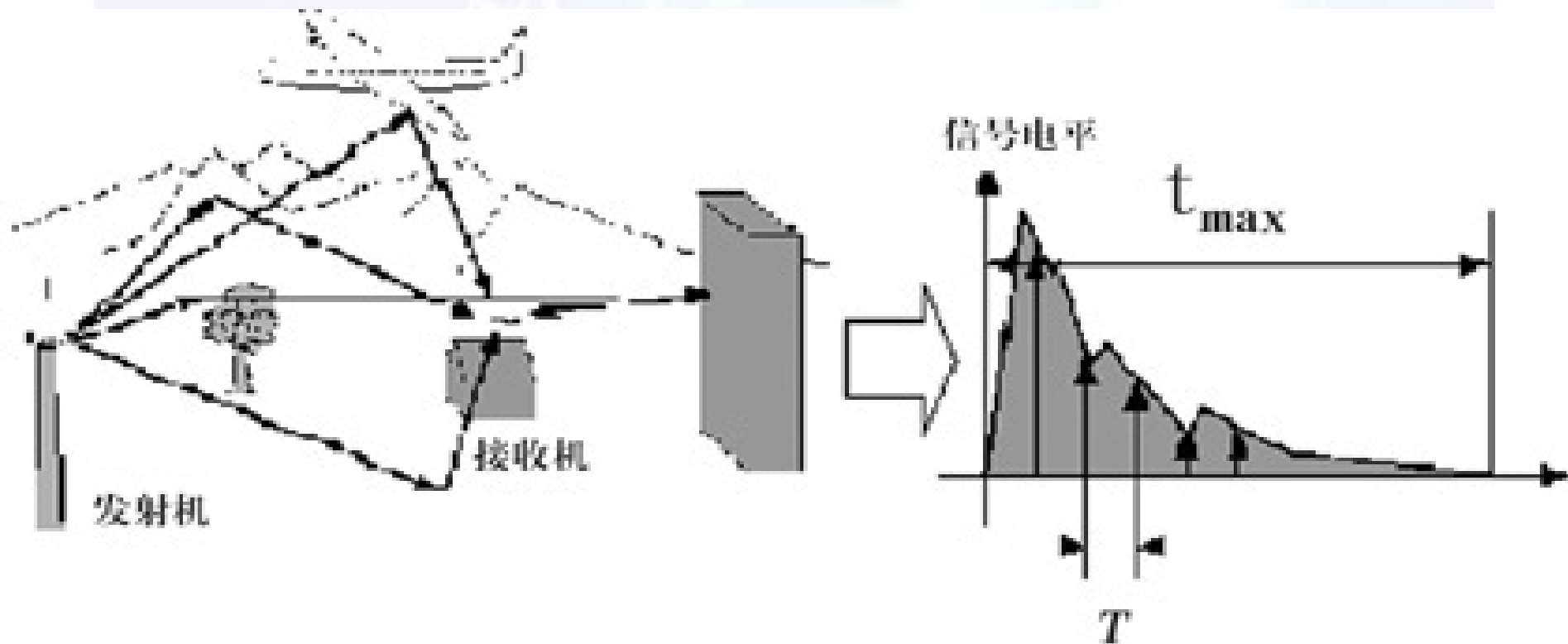
3.7 FFT在OFDM系统中的应用

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

单载波调制：

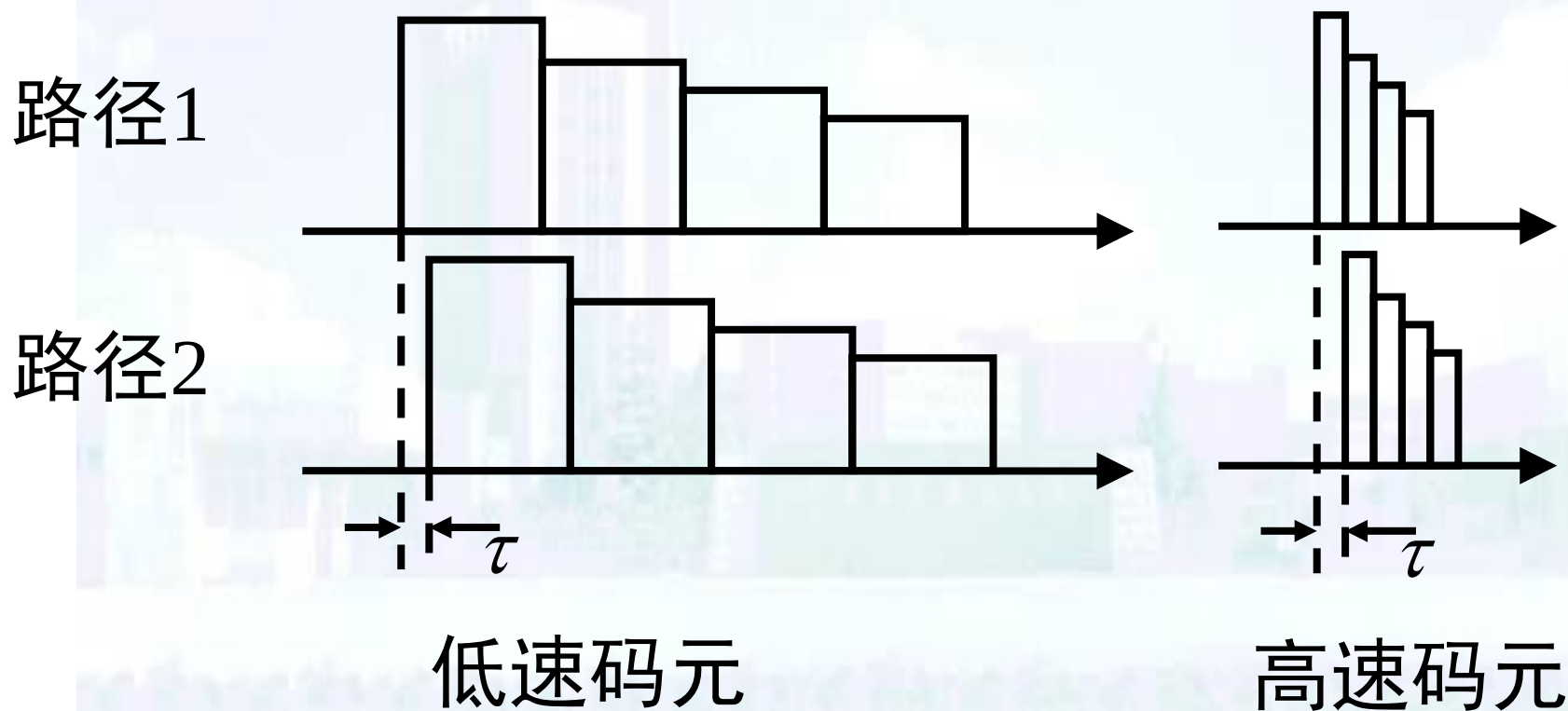
- 时延扩展，产生符号间干扰
- 信号的带宽过宽，频率选择性衰落
- 信道均衡困难

3.7 FFT在OFDM系统中的应用



3.7 FFT在OFDM系统中的应用

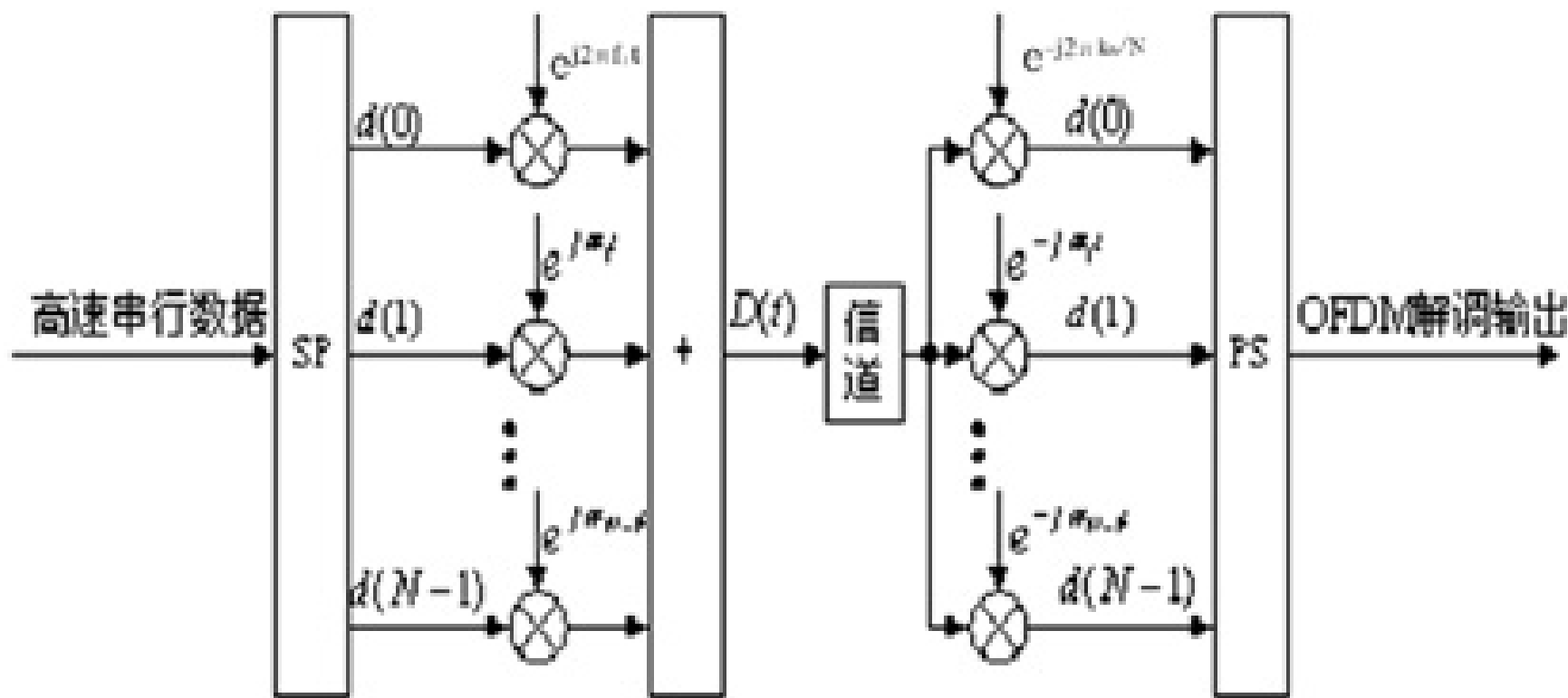
相同多径时延下不同传输速率的符号间干扰对比



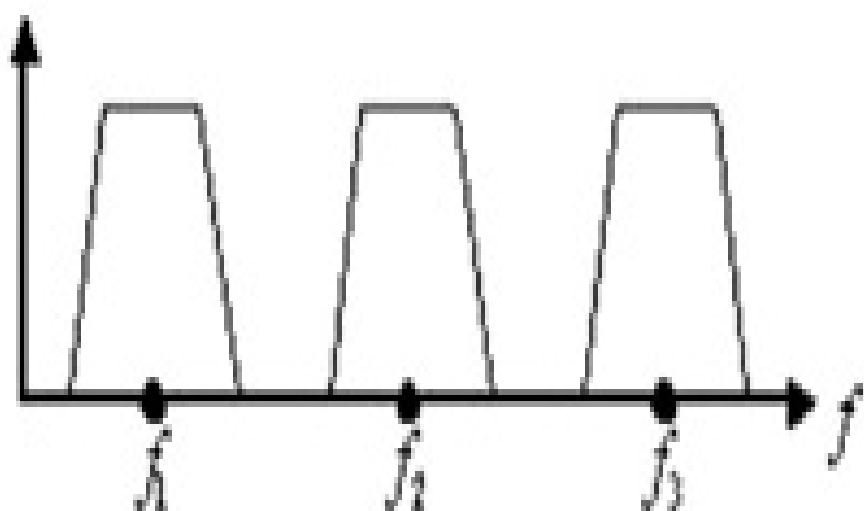
3.7 FFT在OFDM系统中的应用

多载波调制:

- 高速串行码元 $\rightarrow$ 低速并行码元
- 低速并行码元分别调制不同的载波

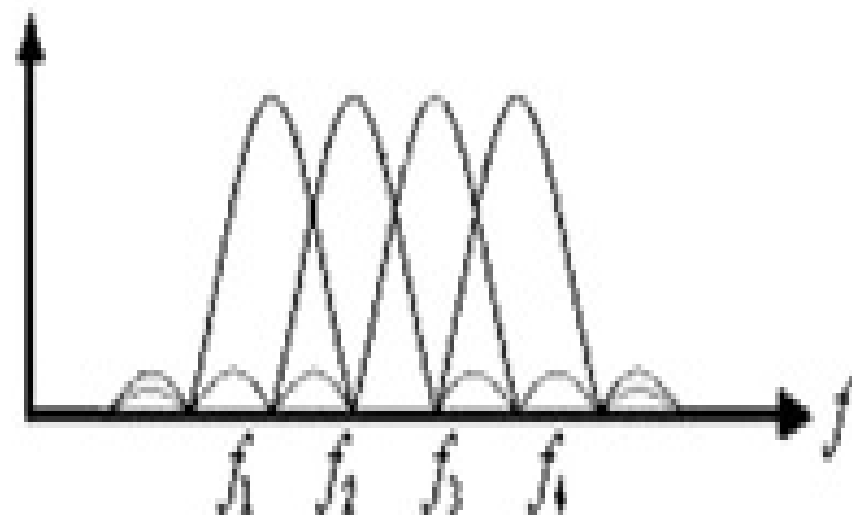


3.7 FFT在OFDM系统中的应用



多载波

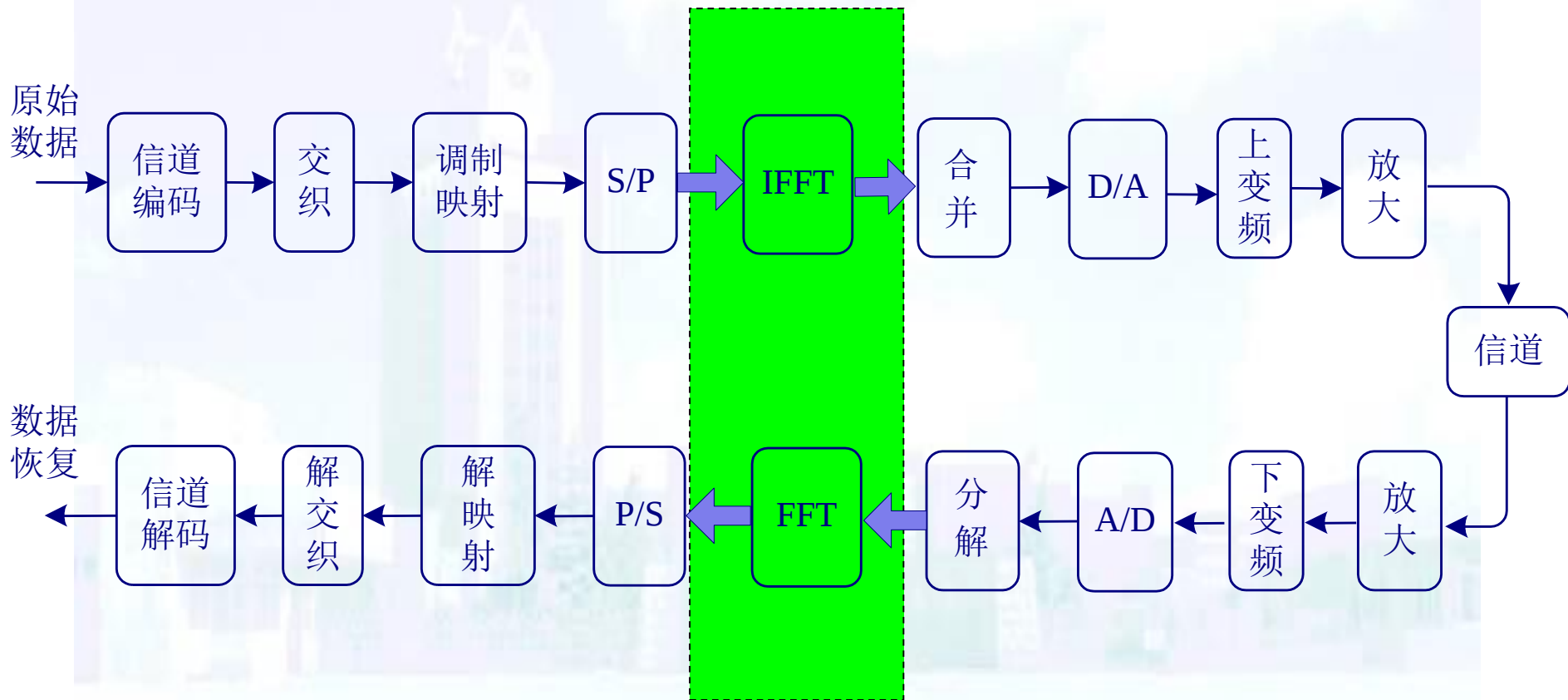
- 频谱效率较低
- 实现困难



OFDM

- 频谱效率较高
- 可用FFT实现

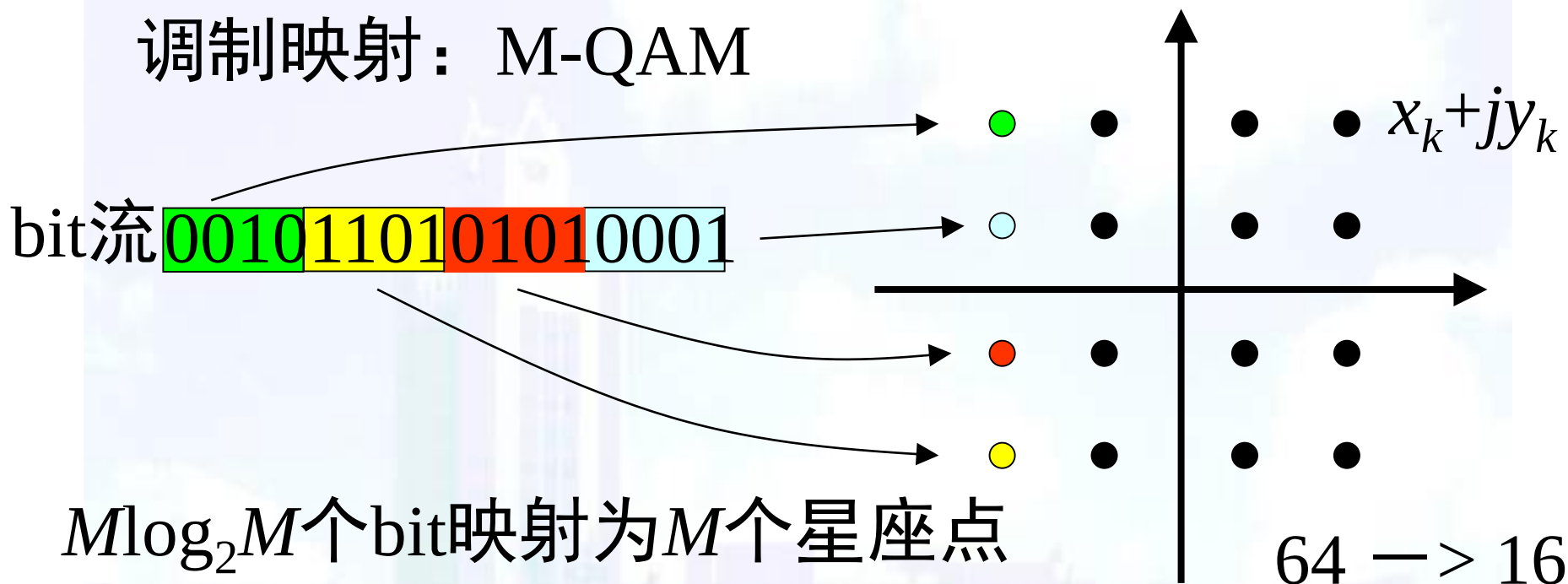
3.7 FFT在OFDM系统中的应用



OFDM调制系统原理

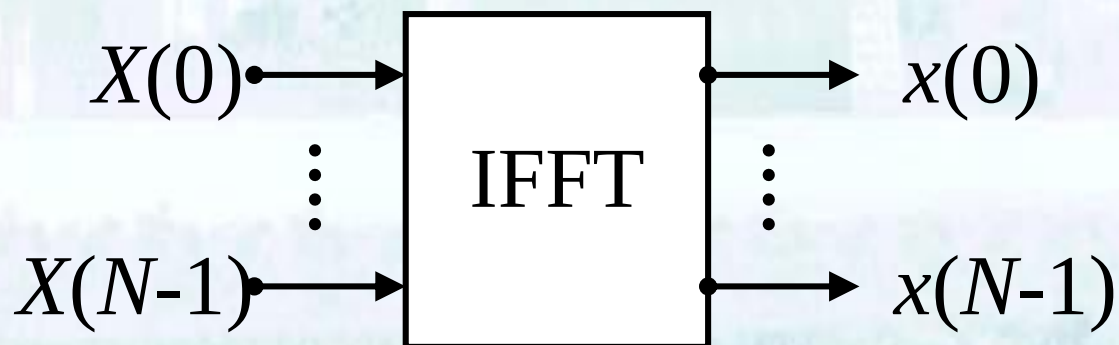
3.7 FFT在OFDM系统中的应用

调制映射：M-QAM



$M \log_2 M$ 个bit映射为 M 个星座点

$$X(k) = x_k + jy_k, \quad k=0, 1, \dots, N-1$$



3.7 FFT在OFDM系统中的应用

数据速率 R (bit/s) , 系统带宽 W (Hz)
则, 调制阶数 $M=2^{R/W}$

T_{FFT} : FFT信元周期, 根据信道特性确定
子载波数 N (FFT的点数) : $N=WT_{FFT}$

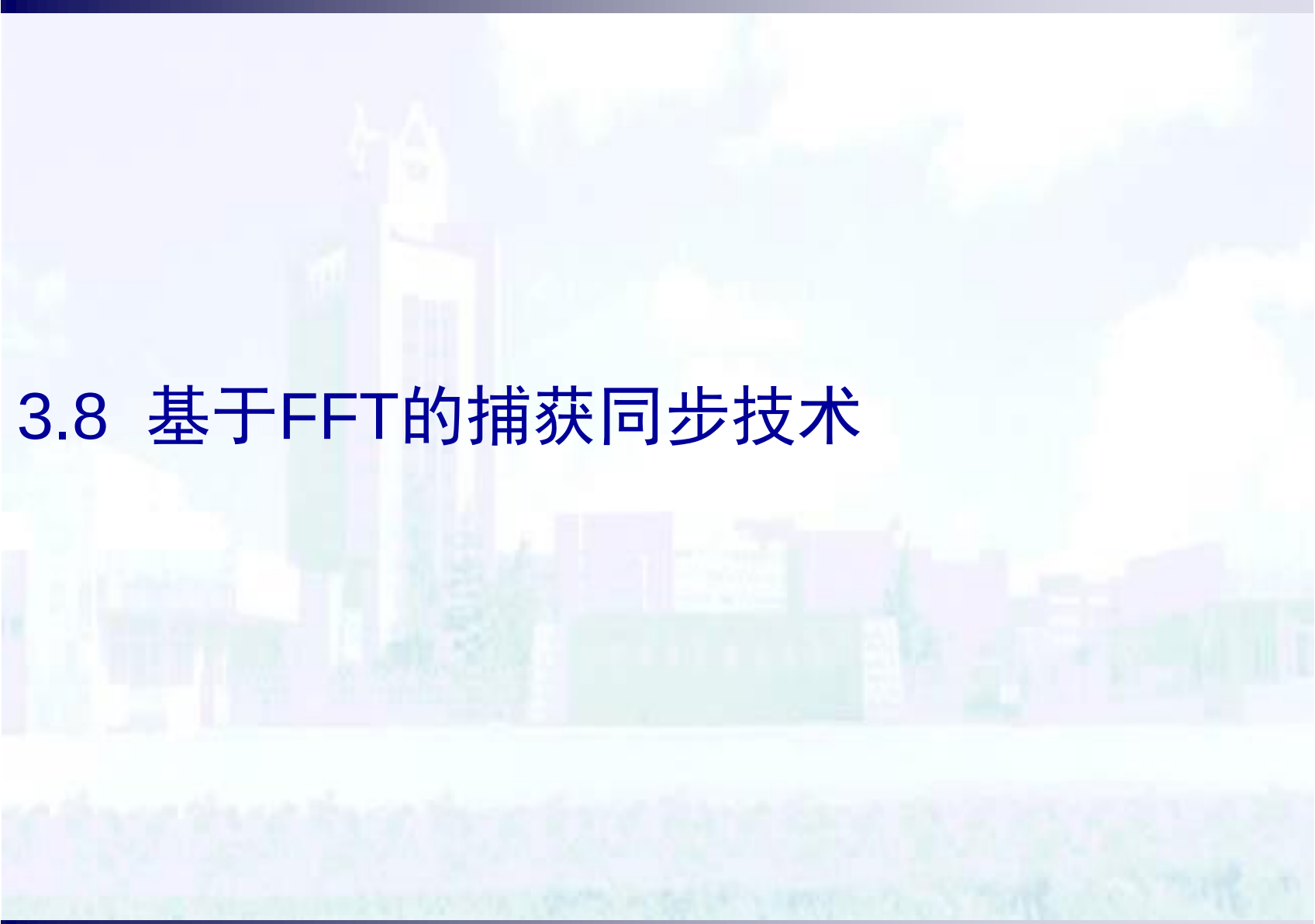
为克服信道多径时延, 在信元之间加保护间隔

3.7 FFT在OFDM系统中的应用

B3G项目系统参数

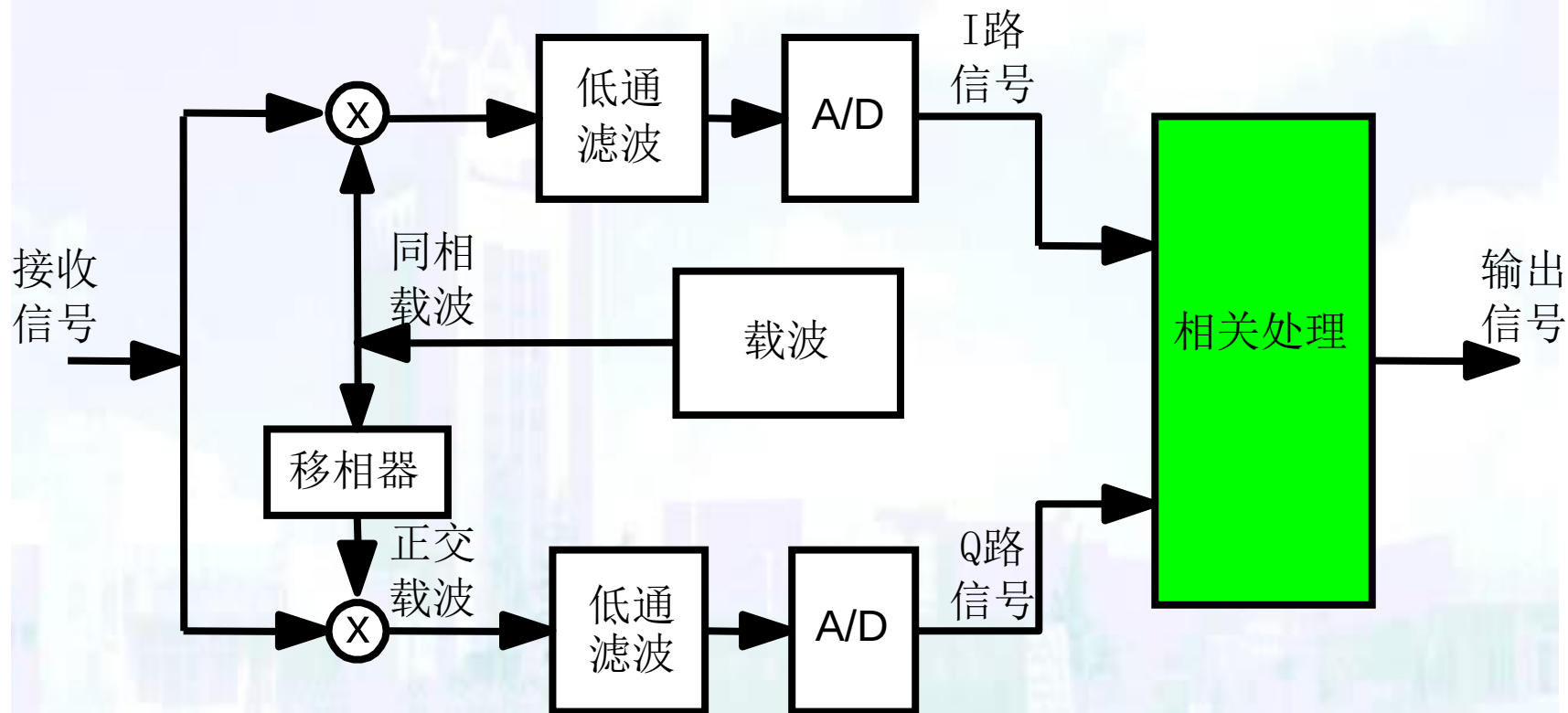
(Beyond 3rd Generation Mobile Communication System)

数据速率	20Mb/s
信道编码	1/2 卷积码
调制方式	4QAM (QPSK)
信道最大时延	10 μ s
保护间隔	12.5 μ s
FFT信元周期	80 μ s
子载波数	2048
子载波间隔	12.5kHz
系统带宽	25.6MHz



3.8 基于FFT的捕获同步技术

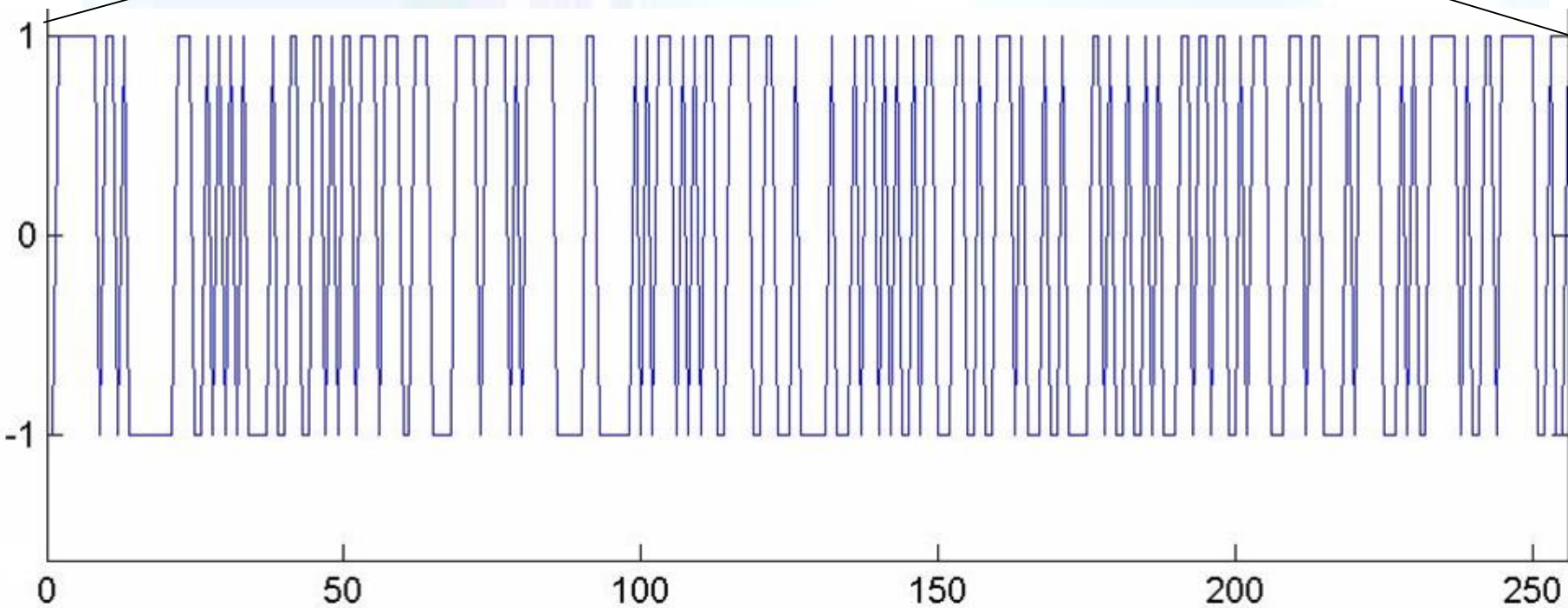
3.8 基于FFT的捕获同步技术



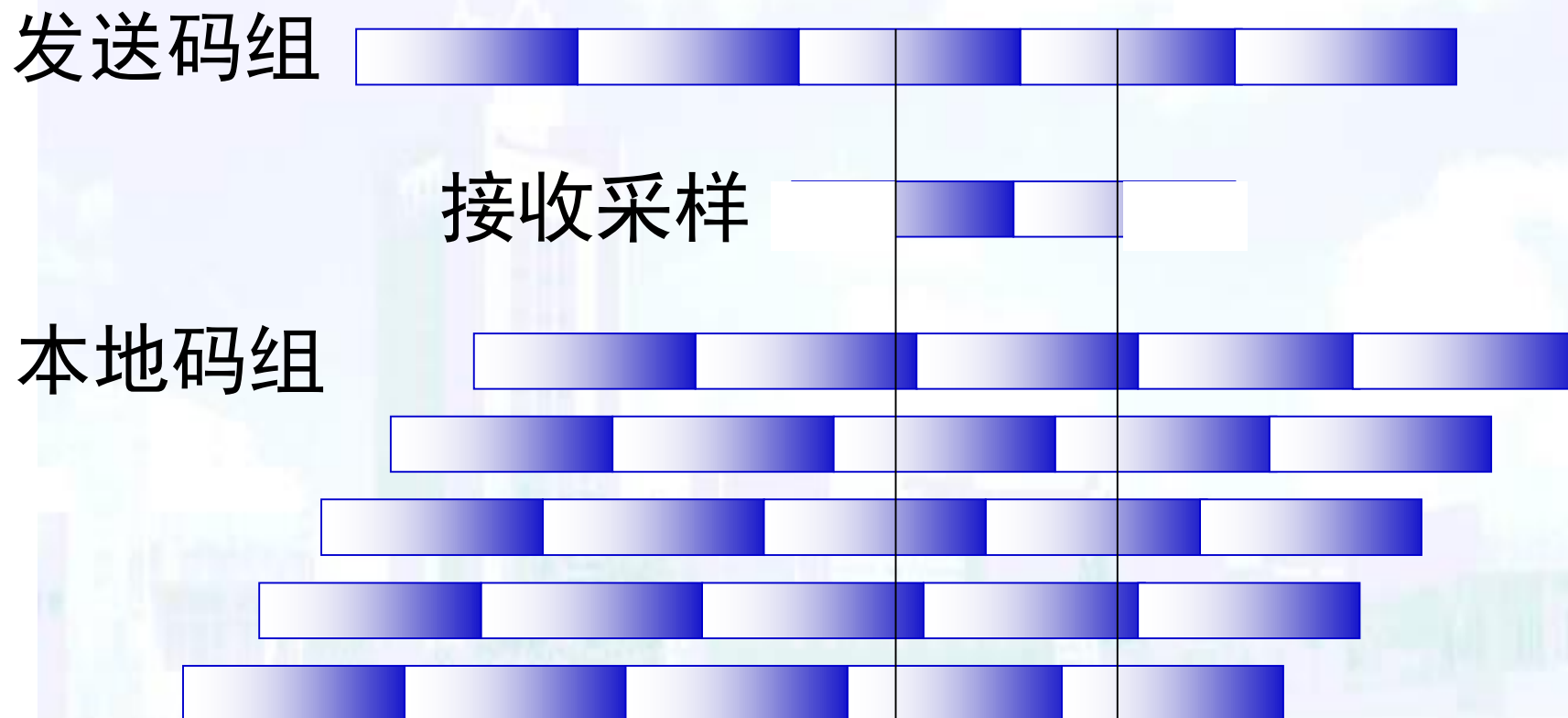
CDMA接收机原理

3.8 基于FFT的捕获同步技术

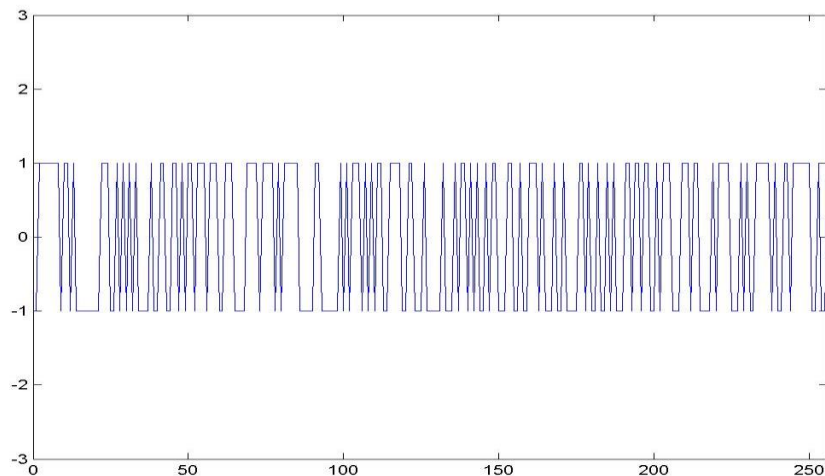
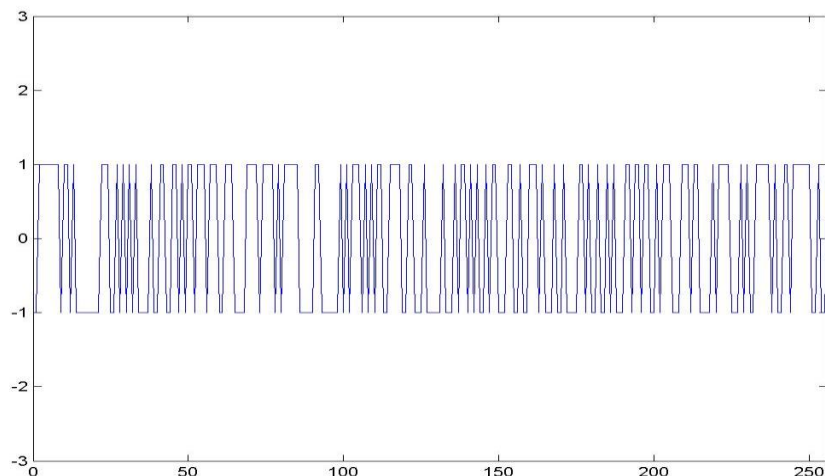
发送端同步码组序列



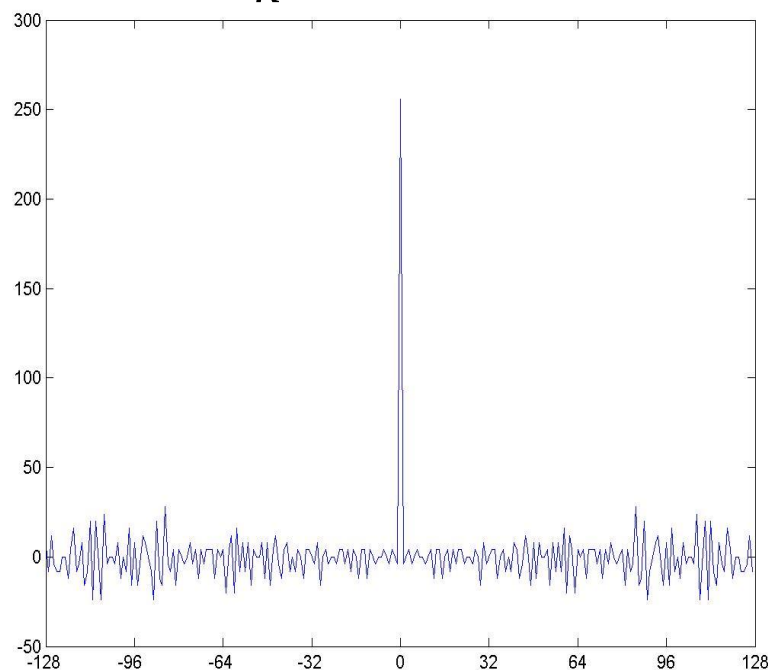
3.8 基于FFT的捕获同步技术



3.8 基于FFT的捕获同步技术

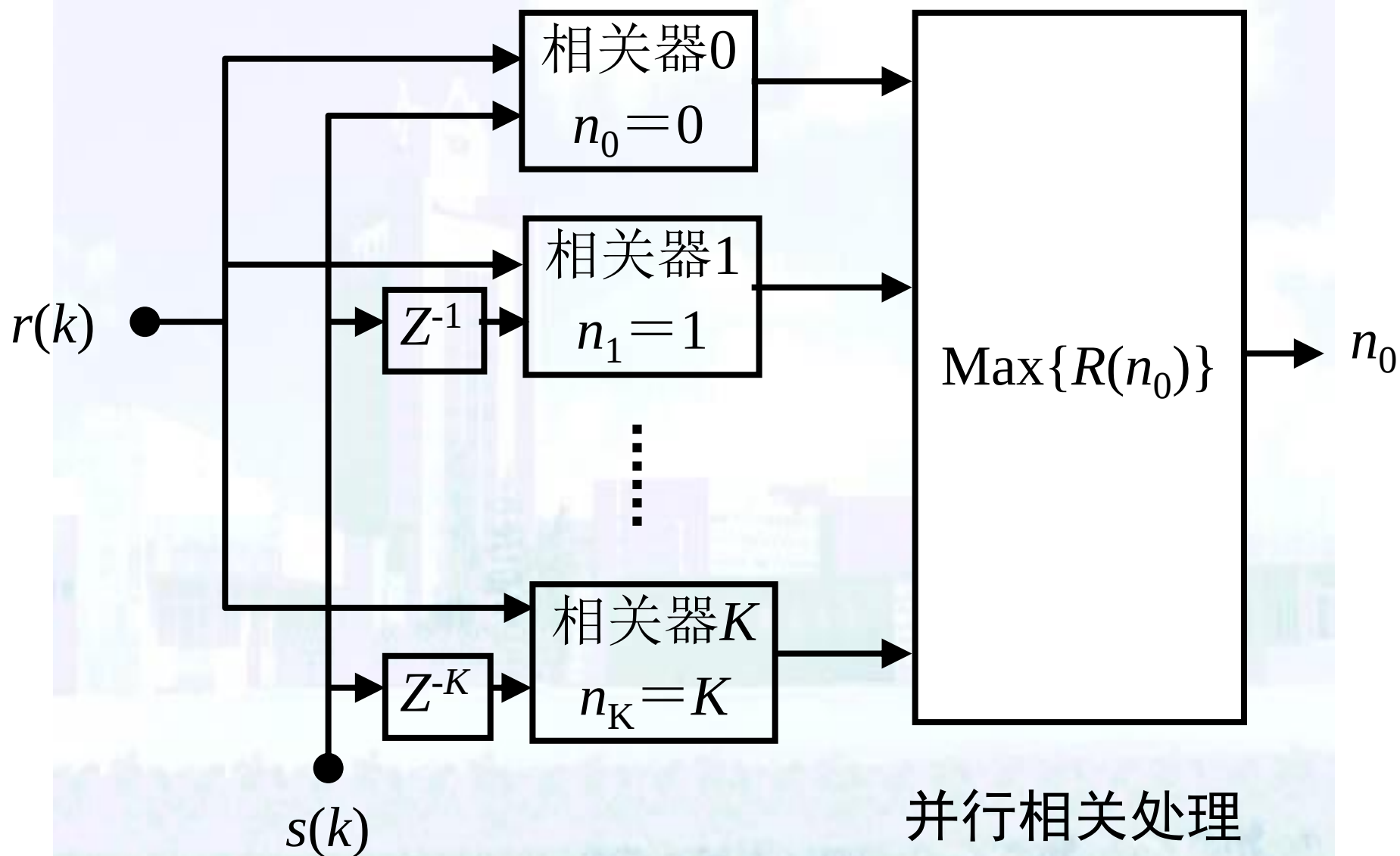


$$R(n_0) = \sum_k r(k)s(k - n_0)$$



相关峰检测

3.8 基于FFT的捕获同步技术



3.8 基于FFT的捕获同步技术

基于FFT的捕获同步原理

