

第四章 快速傅里叶变换

Fast Forurier Transform

4.1

直接计算DFT的问题及改进途径

4.2

基于时间抽取的基-2-FFT快速算法

4.3

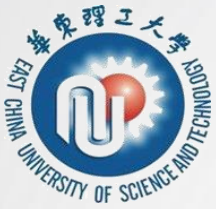
基于频率抽取的基-2-FFT快速算法原理

4.4

快速傅里叶反变换的实现方法

4.5

进一步而减少运算量的措施



第四章 快速傅里叶变换

Fast Forurier Transform

4.4 快速傅里叶反变换的实现方法

华东理工大学信息科学与工程学院 万永菁



4.4 快速傅里叶反变换的实现方法



FFT算法流图也可以用于离散傅里叶逆变换(Inverse Discrete Fourier Transform, 简称IDFT)。

比较DFT和IDFT的运算公式:

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

$$x(n) = \text{IDFT}[X(k)] = \frac{1}{N} \sum_{k=0}^{N-1} X(k)W_N^{-nk} \quad \underline{0 \leq n \leq N-1}$$

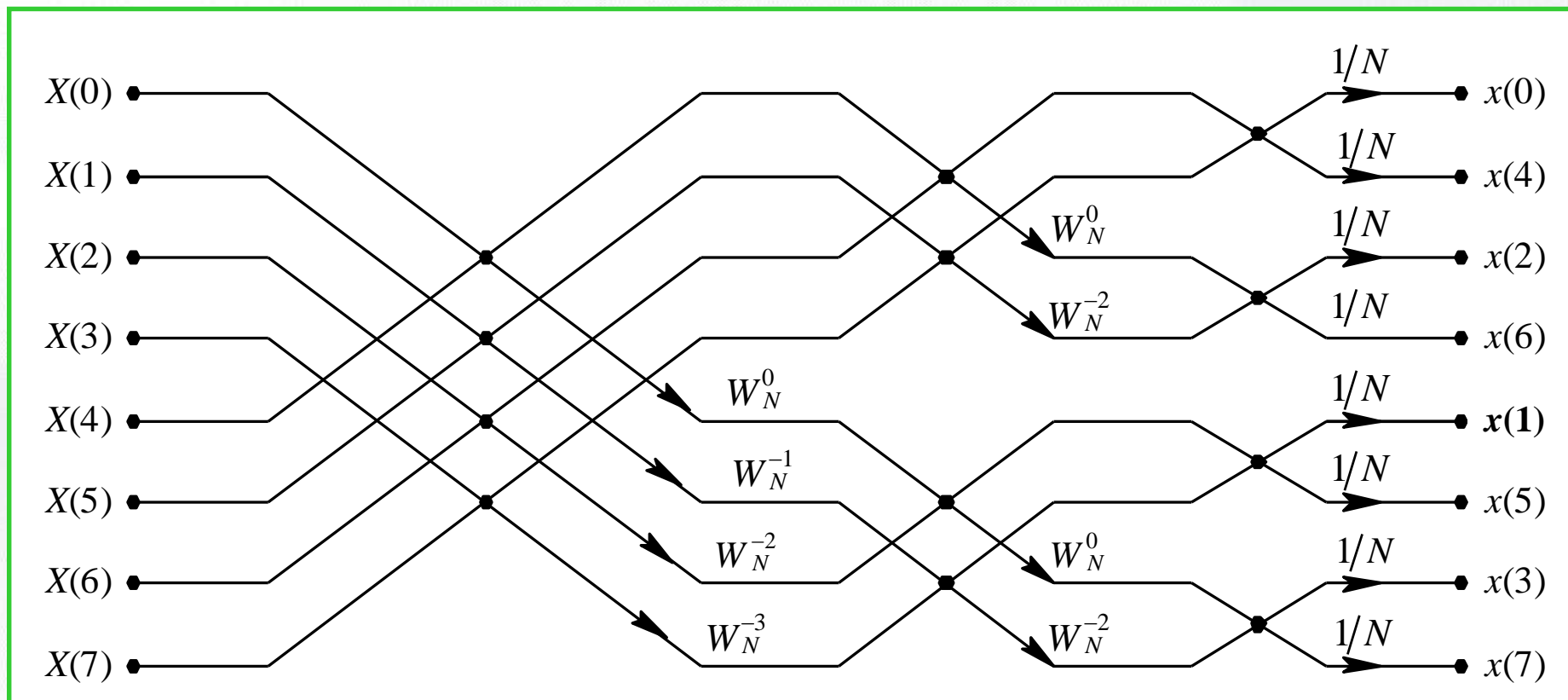
(1) 旋转因子: $W_N^k \rightarrow W_N^{-k}$

(2) 系数: $1/N$, $N = 2^M$

4.4 快速傅里叶反变换的实现方法



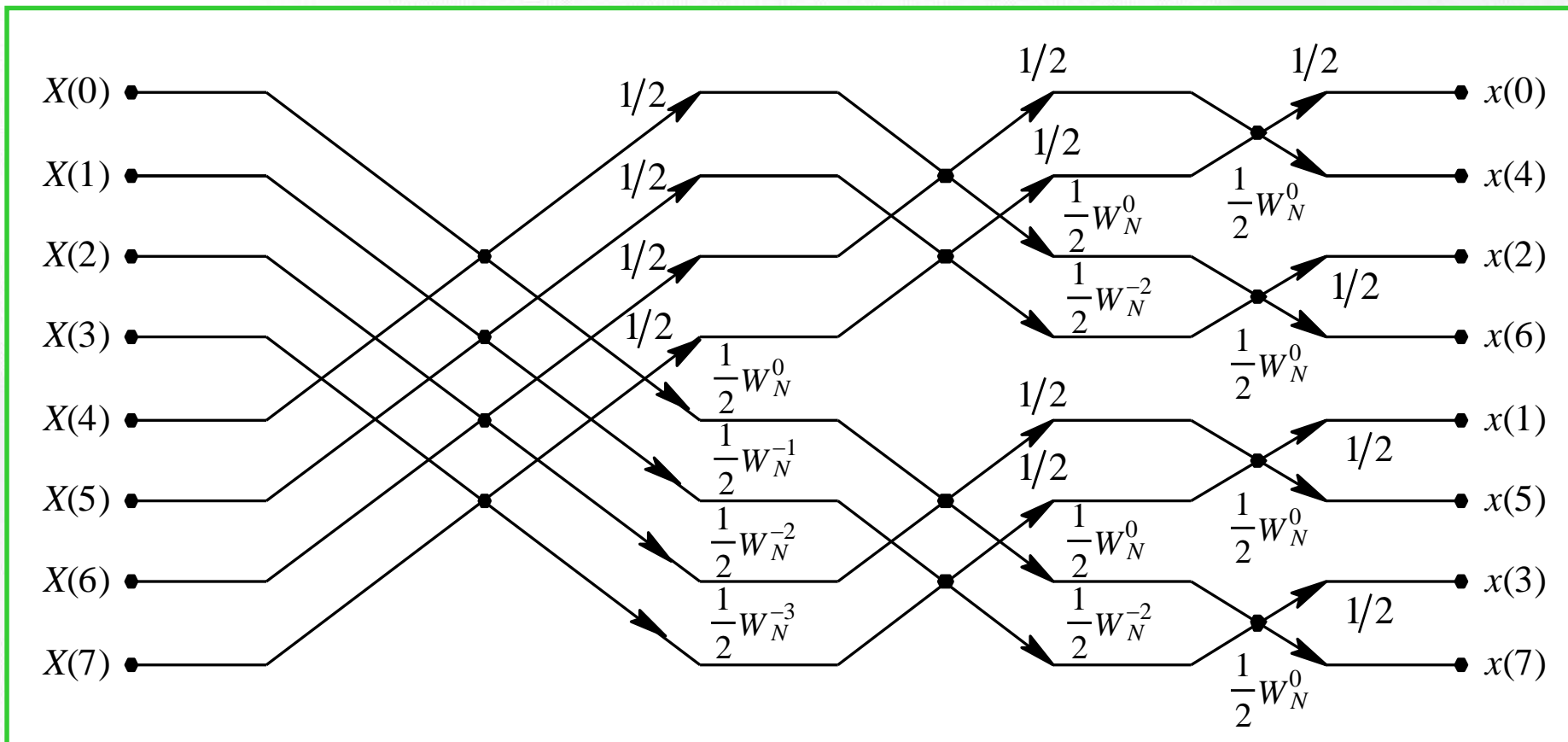
DIT—IFFT运算流图



4.4 快速傅里叶反变换的实现方法



DIT—IFFT运算流图(防止溢出)





4.4 快速傅里叶反变换的实现方法



如果希望直接调用FFT子程序计算IFFT，则可用下面的方法：

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

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

对上式两边再同时取共轭，得：

$$x(n) = \frac{1}{N} \left[\sum_{k=0}^{N-1} X^*(k) W_N^{kn} \right]^* = \frac{1}{N} \left\{ \text{FFT}[X^*(k)] \right\}^*$$



第四章 快速傅里叶变换

Fast Forurier Transform

4.4 快速傅里叶反变换的实现方法

华东理工大学信息科学与工程学院 万永菁

