Digital Signal Processing

College of Communication & Information Engineering
Nanjing University of Posts and Telecommunications

Fall Semester, 2019

JI Wei

Chapter 11 DSP Algorithm Implementation

11.3 Computation of the DFT

本章要点

- 为什么要设计FFT？它是一种新变换吗？
- FFT算法降低计算复杂度的途径？
- FFT算法有几种，各有什么特点？
- DIT、DIF算法基本公式、蝶形图和算法特点

§ 11.3 Computation of the DFT

DFT paris

$$\begin{cases} X(k)=\sum_{n=0}^{N-1} x(n)W_N^{kn} & 0\le k\le N-1 \\ x(n)=\dfrac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-kn} & 0\le n\le N-1 \end{cases}$$

In order to compute the DFT or IDFT of a length-*N* sequence, one needs

***N*²** **complex multiplications** $\mathrm{M}(N)=N^2$

***N*(*N*-1) complex additions** $\mathrm{A}(N)=N(N-1)$

§ 11.3 Computation of the DFT

The complexity of the DFT grows with the square of the signal length.

for example:

N=8 needs 64 complex multiplications.

N=1024 needs 1,048,576 complex multiplications.

This severely limits its practical use for lengthy signals.

In 1965, Cooley and Tukey proposed an efficient algorithm to compute the DFT, that is DIT-FFT.

FFT: fast implementation of the DFT

本章要点

- 为什么要设计FFT？它是一种新变换吗？
- FFT算法降低计算复杂度的途径？
- FFT算法有几种，各有什么特点？
- DIT、DIF算法基本公式、蝶形图和算法特点

§ 11.3 Computation of the DFT

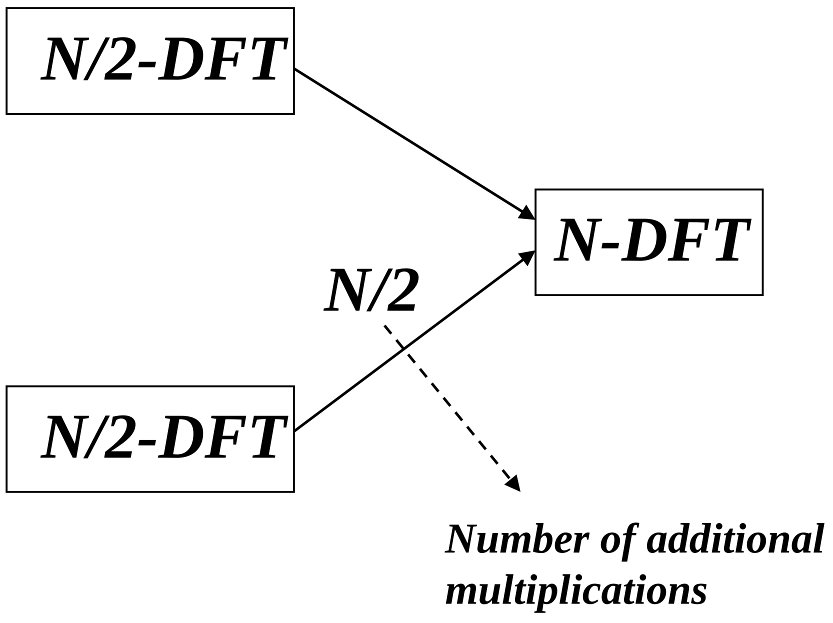
Common principle of the FFT algorithms:

Convert the DFT of a long sequence into the merging (组合) of the DFTs of shorter sequences.

考察逐级分解为短序列DFT后的乘法计算量:

Consider *N*-point DFT of length-*N* sequence when $\boxed{N = 2^B}$, we investigate the cost of multiplications when rebuilding N-point DFT from various length (shorter) DFTs:

Basic merging unit



When *N*-DFT is rebuilt out of two *N/2*-DFTs, the total cost is:

$$2\times\left(\frac{N}{2}\right)^2+\frac{N}{2}=\frac{N^2}{2}+\frac{N}{2}\overset{N大}{\approx}\frac{N^2}{2}$$

（一级分解）

$\frac{N}{2}$: **additional cost in merging**

When each (*N/2*)-DFT is rebuilt out of two *N/4*-DFTs, total cost is:

$$4\times\left(\frac{N}{4}\right)^2+\frac{N}{4}+\frac{N}{4}+\frac{N}{2}=\frac{N^2}{4}+2\times\frac{N}{2}\overset{N大}{\approx}\frac{N^2}{4}$$

（二级分解）

(cost of merging)

When the subdivision process is continued for *m* stages,

Total cost: $$\frac{N^2}{2^m} + m \times \frac{N}{2}$$ **（m级分解）**

If $m = B = \log_2 N$ **, the total cost is:**

$$\frac{N^2}{2^B} + B \times \frac{N}{2} = N + B \times \frac{N}{2} \approx B \times \frac{N}{2} = \boxed{\frac{N}{2}\log_2 N}$$

FFT算法降低计算复杂度的途径

- 方法：

  ① 将长序列的DFT分解成短序列的DFT

  ② 利用W因子的周期性和对称性

§ 11.3.2 Cooley-Tukey FFT Algorithms

Using the properties of the W_N^{nk}

1. $(W_N^{nk})^* = W_N^{-nk}$

2. $W_N^{nk} = W_N^{(n+N)k} = W_N^{n(k+N)}$ $\qquad W_N^{nk+\frac{N}{2}} = -W_N^{nk}$

3. $W_N^{nk} = W_{mN}^{nmk} = W_{N/m}^{nk/m}$

FFT算法有几种?

•Decompose the long sequence into shorten sequence to compute DFT

Decimation in time
Decimation in frequency

本章要点

- 为什么要设计FFT？它是一种新变换吗？
- FFT算法降低计算复杂度的途径？
- FFT算法有几种，各有什么特点？
- DIT、DIF算法基本公式、蝶形图和算法特点

§ 11.3.2 Cooley-Tukey FFT Algorithms

Radix-2 algorithm with decimation in time-DIT FFT

1. Principle of algorithm

Suppose: $x(n)$ length $N=2^L$

split $x(n)$ into two parts $\begin{cases} \text{even-index} & 2n \\ \text{odd-index} & 2n+1 \end{cases}$

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

§ 11.3.2 Cooley-Tukey FFT Algorithms

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

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

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

§ 11.3.2 Cooley-Tukey FFT Algorithms

$$\because W_N^{2nk} = W_{N/2}^{nk}$$

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

N/2 points DFT ***N/2 points DFT***

$$0 \le k \le \frac{N}{2} - 1$$

$X_e(k)$ $X_o(k)$

$$\therefore X(k) = X_e(k) + W_N^k X_o(k) \qquad 0 \le k \le \frac{N}{2} - 1$$

However, we only compute the N/2 points of length-N DFT at beginning, and remind other N/2 points.

§ 11.3.2 Cooley-Tukey FFT Algorithms

$$\begin{cases} X(k) = X_e(k) + W_N^k X_o(k) & 0 \le k \le \frac{N}{2} - 1 \\ X(k+\frac{N}{2}) = X_e(k+\frac{N}{2}) + W_N^{k+\frac{N}{2}} X_o(k+\frac{N}{2}) & \frac{N}{2} \le k + \frac{N}{2} \le N \end{cases}$$

$$\because X_e(k+\frac{N}{2}) = \sum_{n=0}^{N/2-1} x(2n) W_{N/2}^{n(k+N/2)} = \sum_{n=0}^{N/2-1} x(2n) W_{N/2}^{nk} = X_e(k)$$

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

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

§ 11.3.2 Cooley-Tukey FFT Algorithms

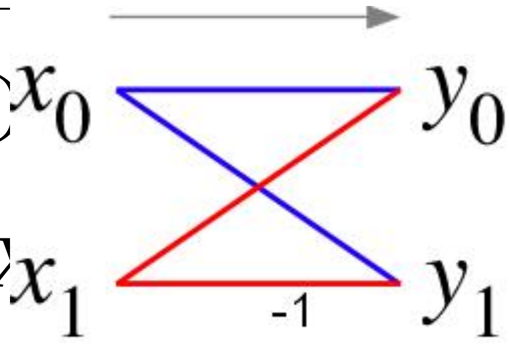
$$\therefore X(k+\frac{N}{2}) = X_e(k+\frac{N}{2}) + W_N^{k+N/2} X_o(k+\frac{N}{2})$$

$$= X_e(k) - W_N^k X_o(k) \qquad 0 \le k \le \frac{N}{2} - 1$$

Now, we obtain the remind *N*/2 points of length-*N* DFT.

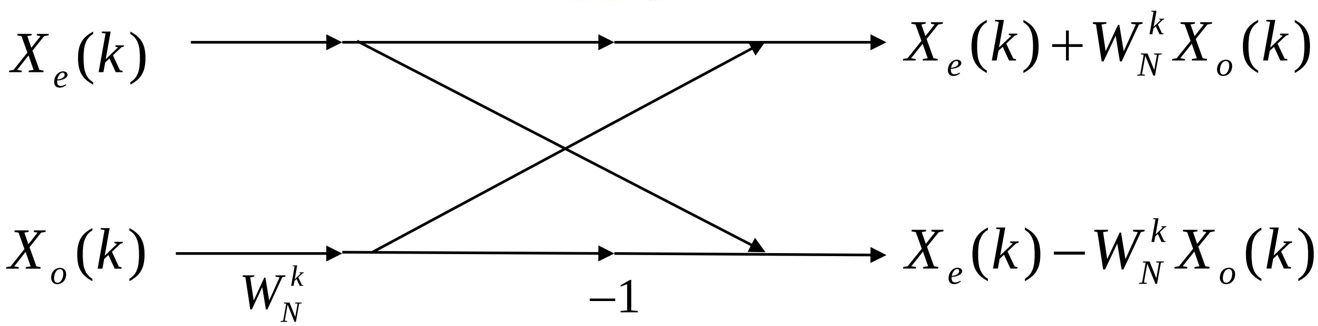
§ 11.3.2 Cooley-Tukey FFT Algorithms

$$\begin{cases} X(k) = X_e(k \\ X(k+\frac{N}{2}) = \end{cases}$$



$$0 \le k \le \frac{N}{2} - 1$$

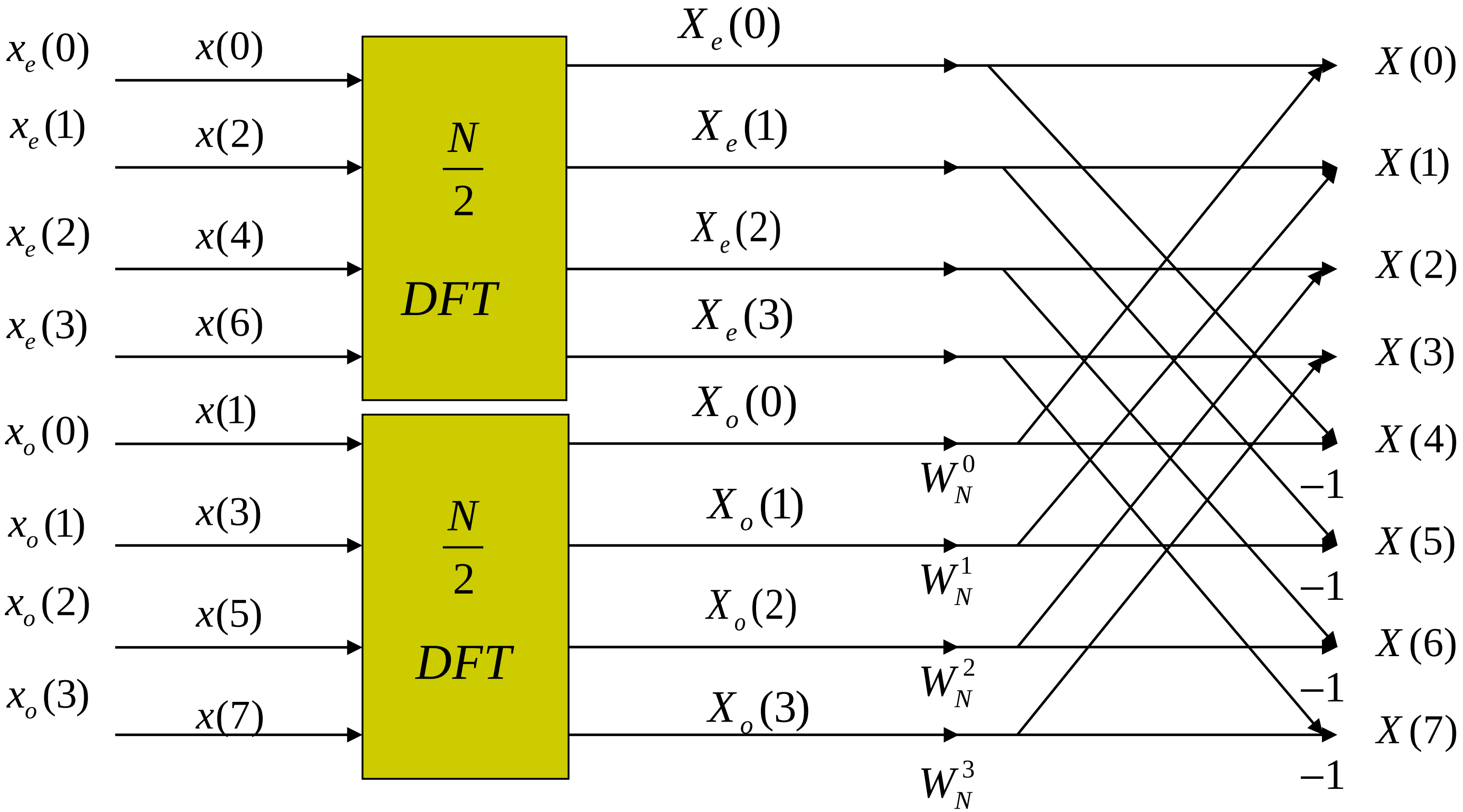
called basic ce **m, and its signal flow graph is**



also called a butterfly

$$\begin{cases} X(k) = X_e(k) + W_N^k X_o(k) \\ X(k + \frac{N}{2}) = X_e(k) - W_N^k X_o(k) \end{cases} \quad 0 \le k \le \frac{N}{2} - 1$$

for example N=8=2³



§ 11.3.2 Cooley-Tukey FFT Algorithms

each butterfly needs

one complex multiplication

two complex additions

so, after one decomposed

N/2 butterflies $\begin{cases} M(N) = (\frac{N}{2}) \\ A(N) = 2 \times \frac{N}{2} \end{cases}$

2 DFT with N/2 points $\begin{cases} M(N) = 2 \times (\frac{N}{2})^2 \\ A(N) = 2 \times \frac{N}{2}(\frac{N}{2} - 1) \end{cases}$

$$M(N) = \frac{N^2}{2}$$

$$A(N) = N(\frac{N}{2} - 1)$$

its number of computation is almost half of direct DFT.

§ 11.3.2 Cooley-Tukey FFT Algorithms

Now, $x_e(n)$ and $x_o(n)$ is length N/2 sequences, and N/2 also even， then they can be splited into even-index and odd-index parts furthermore

$$x_e(n) \longrightarrow \begin{cases} x_{ee}(n) = x_e(2n) \\ x_{eo}(n) = x_e(2n+1) \end{cases}$$

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

$$x_o(n) \longrightarrow \begin{cases} x_{oe}(n) = x_o(2n) \\ x_{oo}(n) = x_o(2n+1) \end{cases}$$

§ 11.3.2 Cooley-Tukey FFT Algorithms

then, like the first decomposing

$$\begin{cases} X_e(k) = X_{ee}(k) + W_{N/2}^k X_{eo}(k) \\ X_e(k+\frac{N}{4}) = X_{ee}(k) - W_{N/2}^k X_{eo}(k) \end{cases} \qquad 0 \le k \le N/4-1$$

and

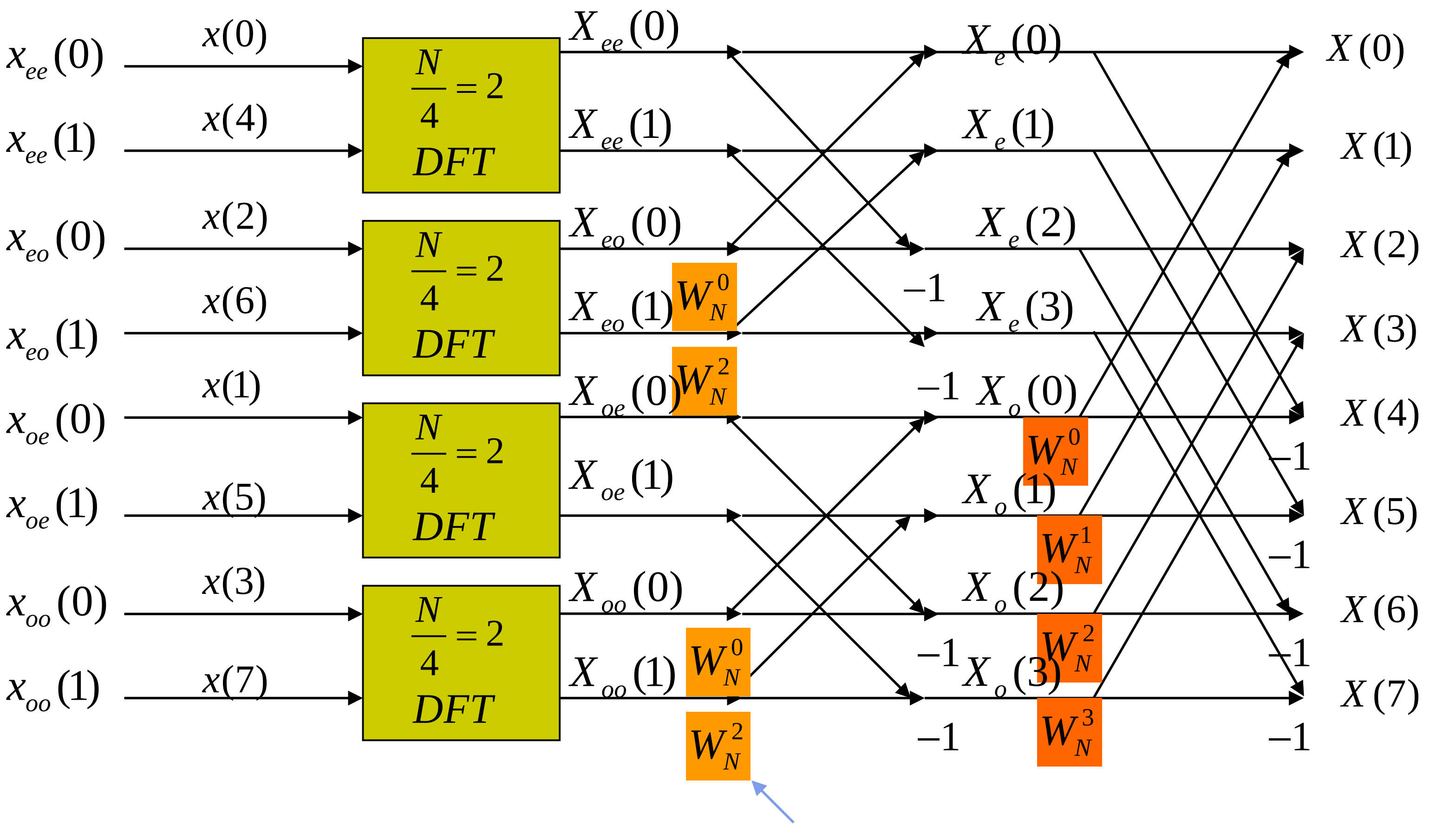
$$\begin{cases} X_o(k) = X_{oe}(k) + W_{N/2}^k X_{oo}(k) \\ X_o(k+\frac{N}{4}) = X_{oe}(k) - W_{N/2}^k X_{oo}(k) \end{cases} \qquad 0 \le k \le N/4-1$$

$where:$

$$x_e(2n) \leftrightarrow X_{ee}(k) \quad x_e(2n+1) \leftrightarrow X_{eo}(k)$$
$$x_o(2n) \leftrightarrow X_{oe}(k) \quad x_o(2n+1) \leftrightarrow X_{oo}(k)$$
$$for \begin{cases} 0 \le n \le N/4-1 \\ 0 \le k \le N/4-1 \end{cases}$$

§ 11.3.2 Cooley-Tukey FFT Algorithms

$N=8=2^3$, and after second decomposing



here W_N, different from $W_{N/2}$ in teaching book

§ 11.3.2 Cooley-Tukey FFT Algorithms

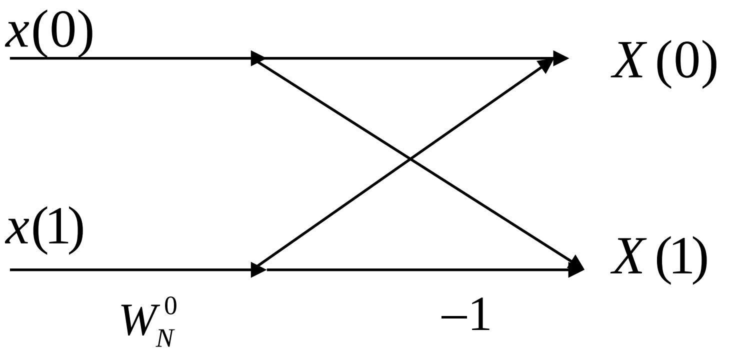
Resembly, we can decompose it to length-2 DFT at last

$$X(k) = \sum_{n=0}^{1} x(n) W_2^{nk} = x(0) W_2^{0k} + x(n) W_2^{k}$$

$$\begin{cases} X(0) = x(0) + x(1) W_2^0 = x(0) + x(1) \\ X(1) = x(0) + x(1) W_2^1 = x(0) - x(1) \end{cases}$$

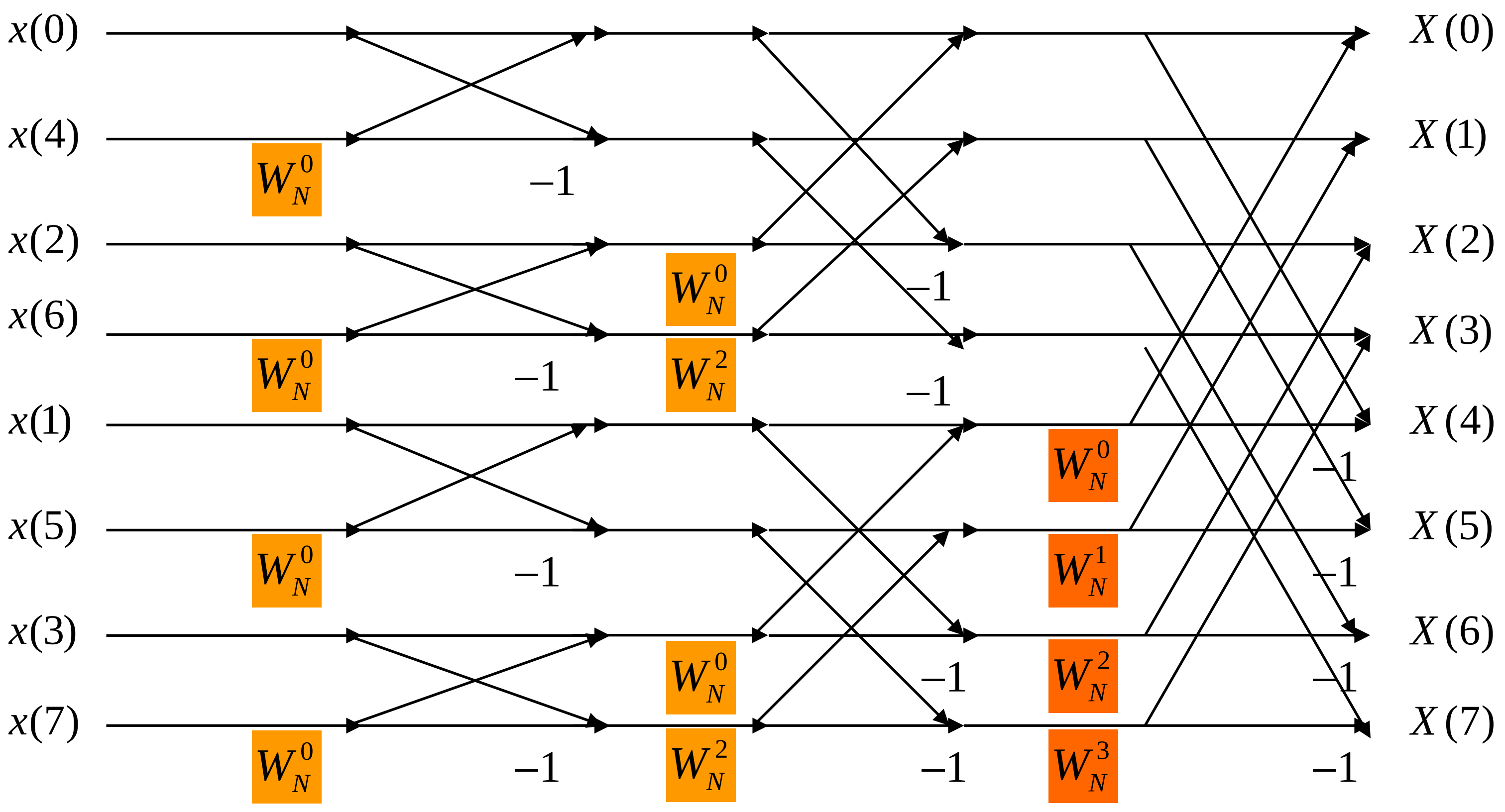
expressed as

$$\begin{cases} X(0) = x(0) + W_N^0 x(1) \\ X(1) = x(0) - W_N^0 x(1) \end{cases}$$



§ 11.3.2 Cooley-Tukey FFT Algorithms

as the example above N=8=2³, at last



§ 11.3.2 Cooley-Tukey FFT Algorithms

2. Number of computation

if $N=2^L$, then there are L levels butterfly cells

per butterfly $\left\{\begin{array}{l}\text{1 complex multiplication} \\ \text{2 complex additions}\end{array}\right.$

N/2 butterflies per level

total L levels to complete length-N DFT

$\Rightarrow$

$$\begin{cases} \mathfrak{M}_F(N) = 1 \cdot \dfrac{N}{2} \cdot L = \dfrac{N}{2} \cdot \log_2 N \\ \mathfrak{A}_F(N) = 2 \cdot \dfrac{N}{2} \cdot L = N \cdot \log_2 N \end{cases}$$

§ 11.3.2 Cooley-Tukey FFT Algorithms

since multiplication costs far more than addition, so we only take care of multiplication

$$\frac{DFT}{FFT} = \frac{N^2}{\frac{N}{2} \cdot \log_2 N} = \frac{2N}{\log_2 N}$$

N	N^2	$N/2\log_2 N$	DFT/FFT
2	4	1	4
8	64	12	5.4
512	262144	2304	113.8
…	…	…	…

§ 11.3.2 Cooley-Tukey FFT Algorithms

3. Features of DIT-FFT

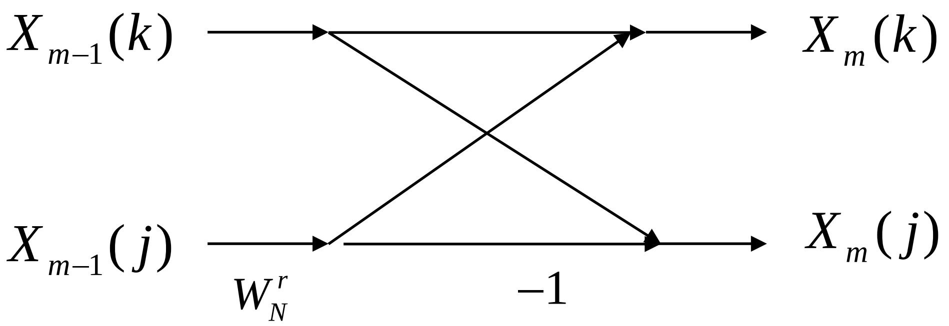
① 'in place' computations

from the basic cell

$$X_m(k) = X_{m-1}(k) + X_{m-1}(j)W_N^r$$

$$X_m(j) = X_{m-1}(k) - X_{m-1}(j)W_N^r$$

and signal flow graph



§ 11.3.2 Cooley-Tukey FFT Algorithms

we can see that the nodes of one section of the graph depend only on nodes of the previous section of the graph.

therefore, once computed, the values of $X_m(k)$ **and** $X_m(j)$ **can be stored in the same place as** $X_{m-1}(k)$ **and** $X_{m-1}(j)$ **.**

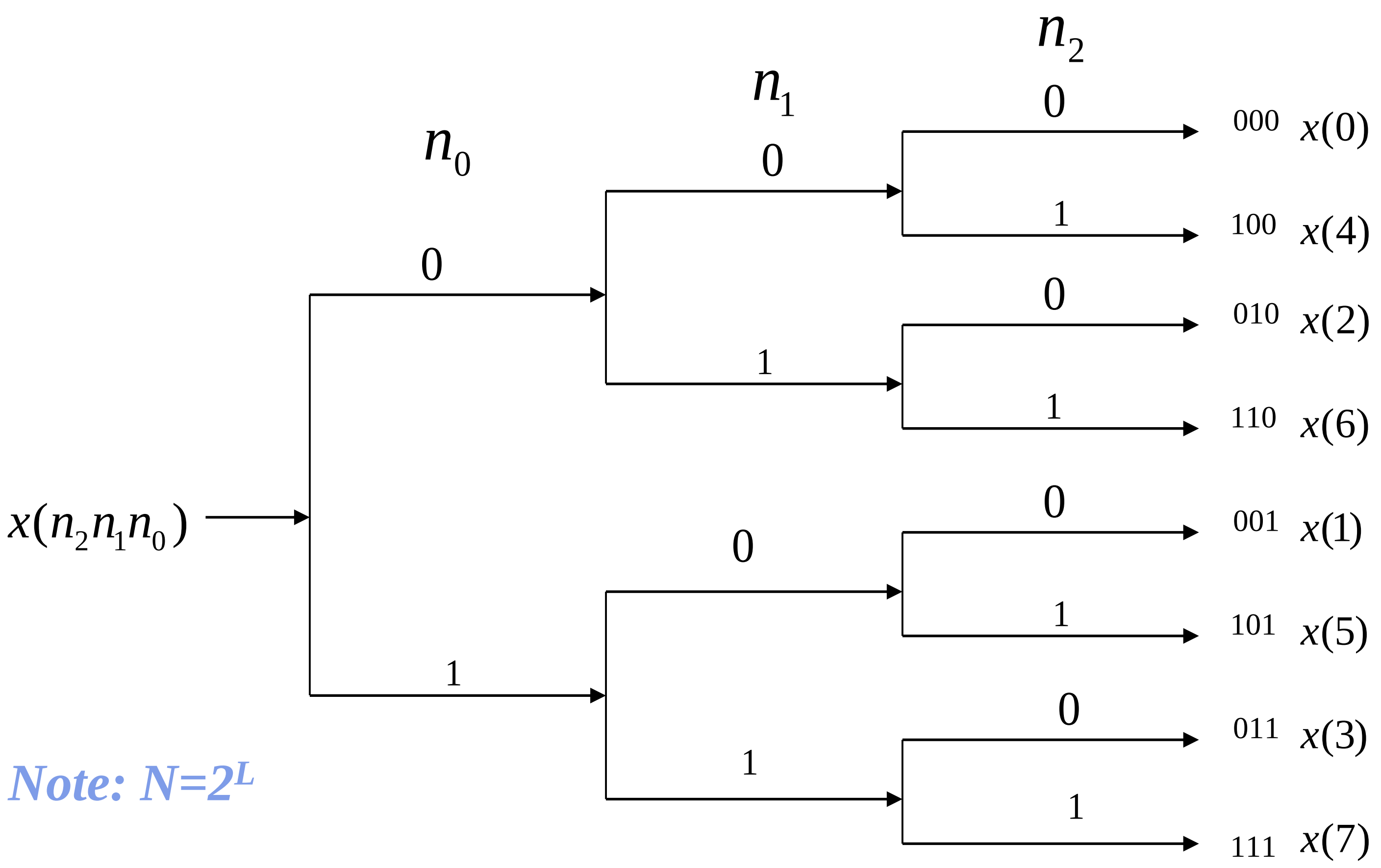
'in place' computation could save storage, and reduce the cost.

Length-N DFT only need

$$\begin{cases} \text{N for } x(n) & n=0,1,\ldots,N-1 \\ \text{N/2 for } W_N^r & r=0,1,\ldots,N/2-1 \end{cases}$$

§ 11.3.2 Cooley-Tukey FFT Algorithms

② bit-reversed order index



Note: $N=2^L$

§ 11.3.2 Cooley-Tukey FFT Algorithms

② bit-reversed order index

Normal order (decimal)	Normal order (binary)	Bit-reversed order (binary)	Bit-reversed order (decimal)
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

§ 11.3.2 Cooley-Tukey FFT Algorithms

③ Distance between nodes of butterfly

N=8 for example

level	distance
1	1
2	2
3	4

conclusion : m-th level, distance is 2^{m-1}

§ 11.3.2 Cooley-Tukey FFT Algorithms

④ Determination for W_N^r

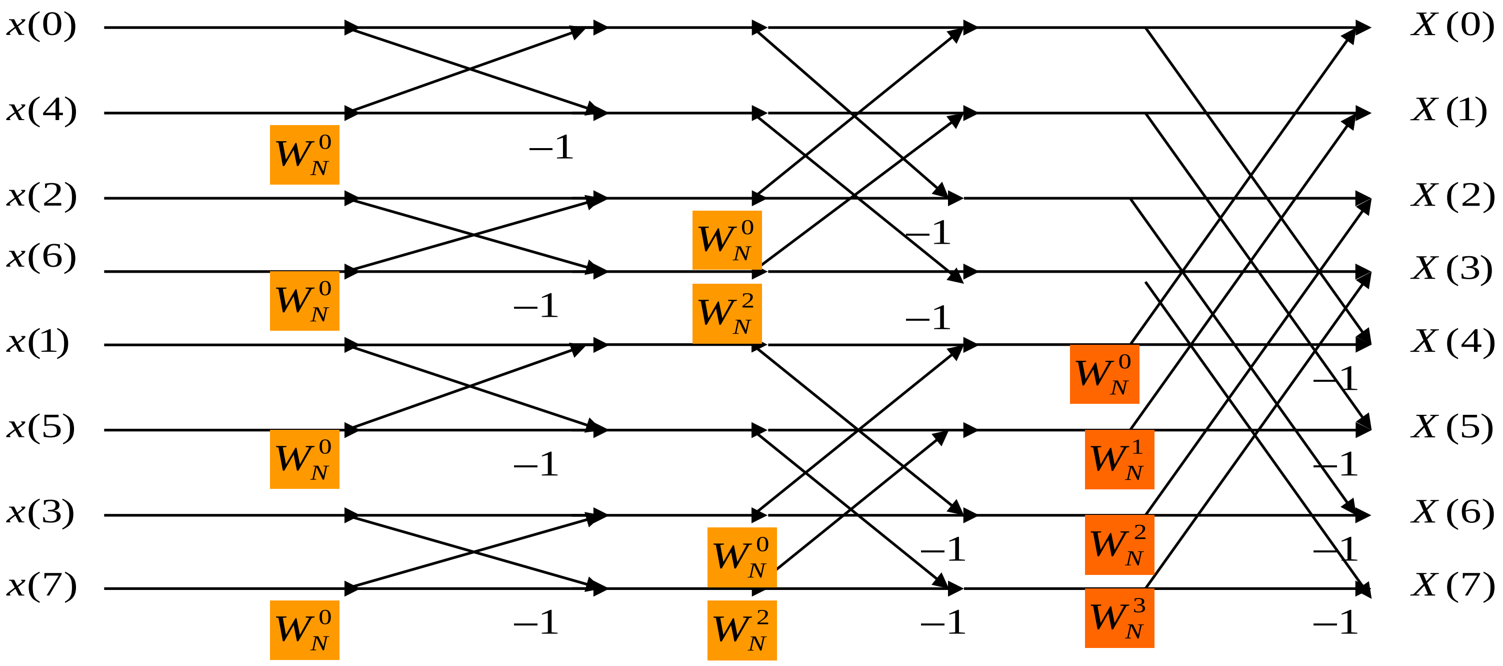
m-th level DIT butterfly could be expressed as

$$X_m(k) = X_{m-1}(k) + X_{m-1}(k+2^{m-1})W_N^r$$

$$X_m(k+2^{m-1}) = X_{m-1}(k) - X_{m-1}(k+2^{m-1})W_N^r$$

r **equals to the** ***k*** **left shift （*L-m*） bits, and zero padded at right.**

§ 11.3.2 Cooley-Tukey FFT Algorithms



§ 11.3.2 Cooley-Tukey FFT Algorithms

Radix-2 algorithm with decimation in frequency

Suppose: $x(n)$ **length** $N=2^L$

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

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

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

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

§ 11.3.2 Cooley-Tukey FFT Algorithms

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

$$\because \quad W_N^{\frac{N}{2}} = -1 \qquad \therefore W_N^{\frac{N}{2}k} = (-1)^k$$

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

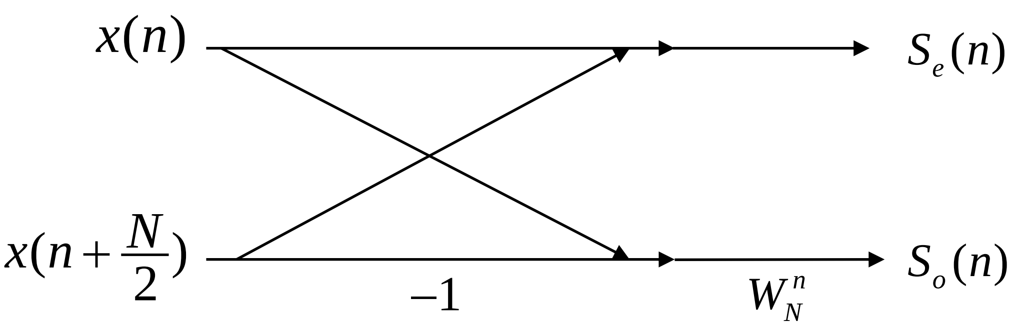
$$\because \quad (-1)^k = \begin{cases} 1 & for\ k\ is\ even \\ -1 & for\ k\ is\ odd \end{cases}$$

§ 11.3.2 Cooley-Tukey FFT Algorithms

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

$$for \quad 0 \le l \le \frac{N}{2} - 1$$

$$make \begin{cases} S_e(n) = x(n) + x(n+\frac{N}{2}) \\ S_o(n) = \left[x(n) - x(n+\frac{N}{2})\right] W_N^{n} \end{cases}$$

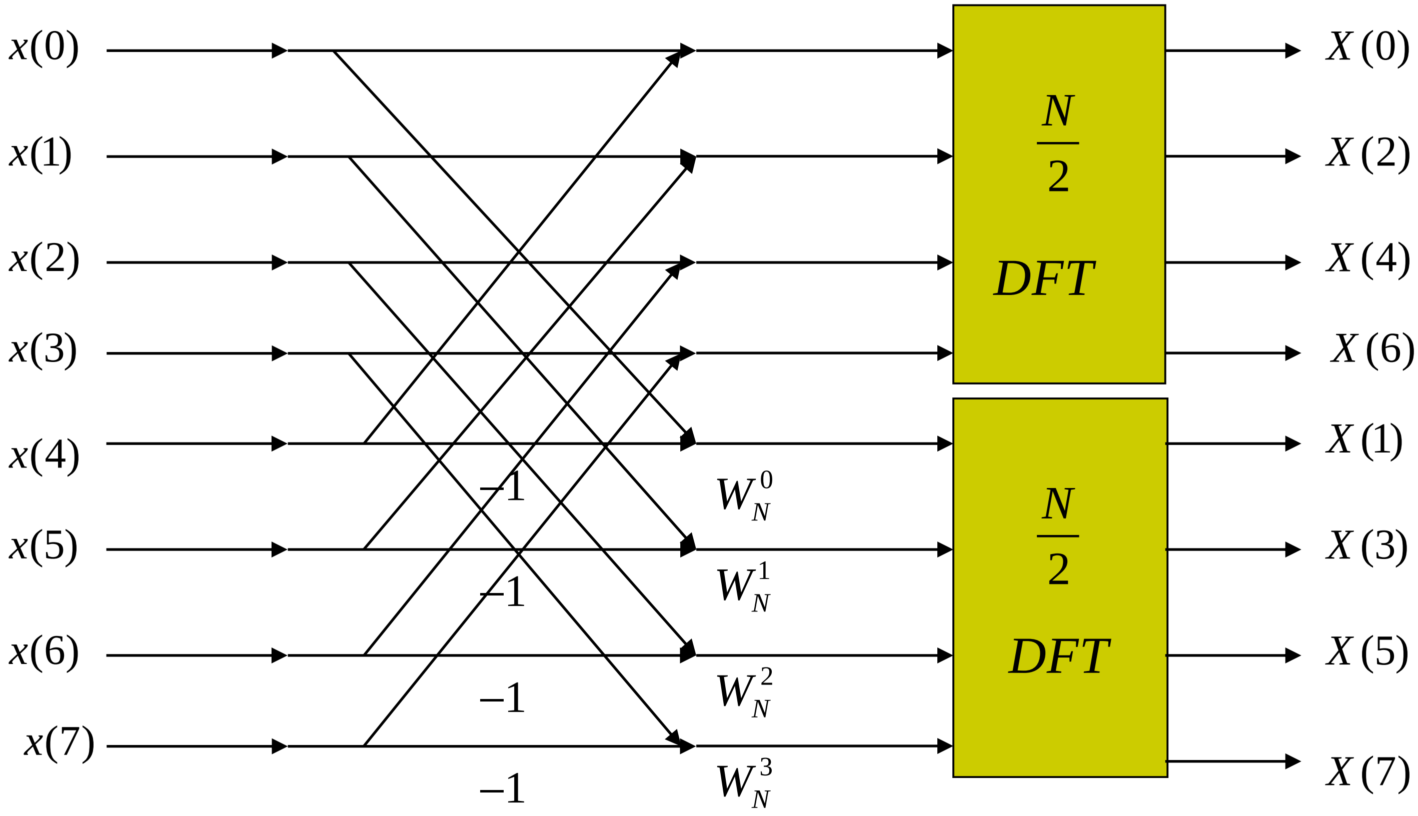


- ***also a butterfly basic cell***
- ***one complex multiplication two complex additions***

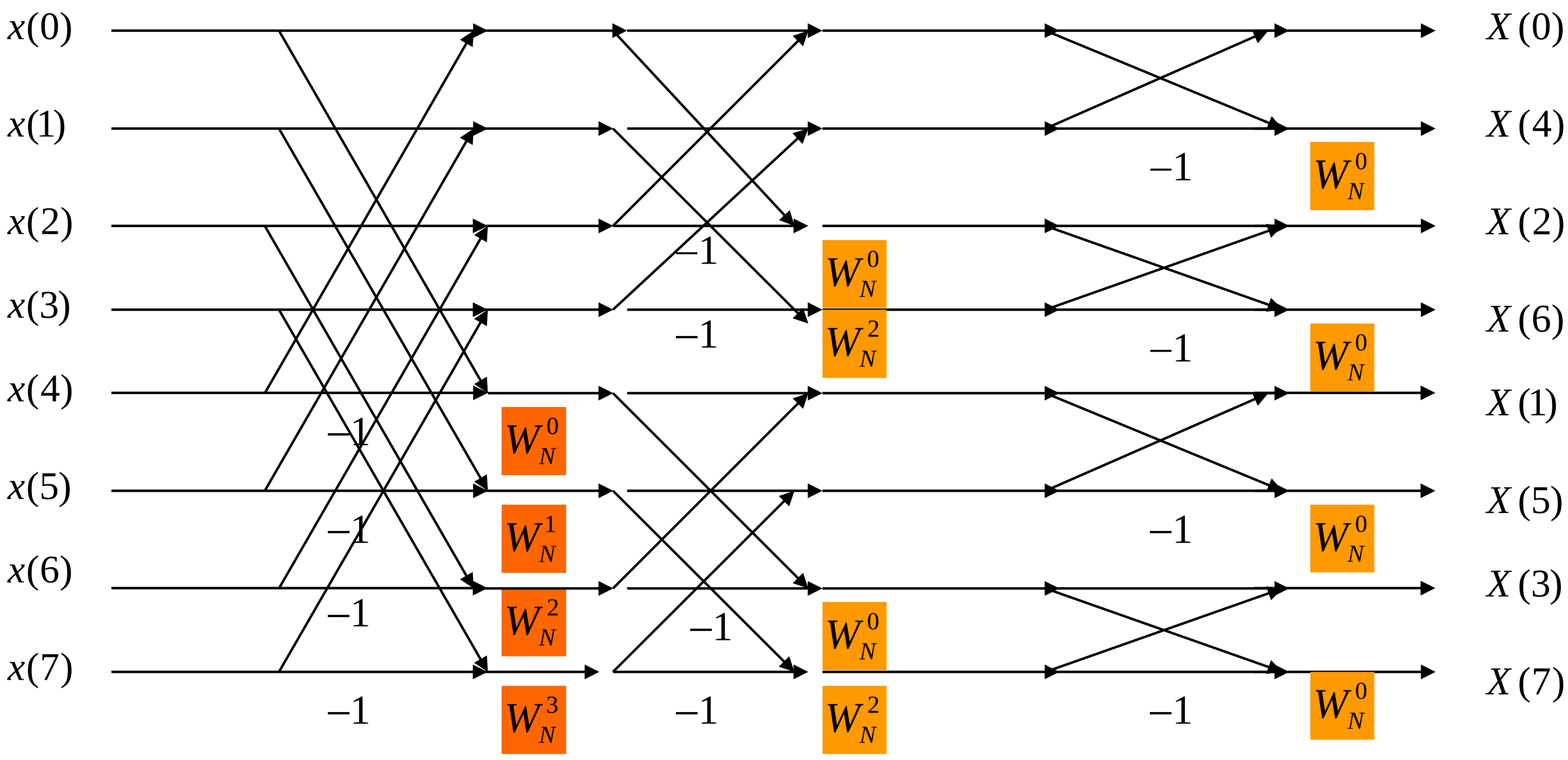
Note: DIF VS DIT

§ 11.3.2 Cooley-Tukey FFT Algorithms

For example N=8



§ 11.3.2 Cooley-Tukey FFT Algorithms



§ 11.3.2 Cooley-Tukey FFT Algorithms

2. Number of computation

if $N=2^L$, then there are L levels butterfly cells

per butterfly { **1 complex multiplication**; **2 complex additions** }

N/2 butterflies per level

total L levels to complete length-N DFT $\Rightarrow$

$$\begin{cases} \mathfrak{M}_F(N) = 1 \cdot \dfrac{N}{2} \cdot L = \dfrac{N}{2} \cdot \log_2 N \\ \mathfrak{A}_F(N) = 2 \cdot \dfrac{N}{2} \cdot L = N \cdot \log_2 N \end{cases}$$

§ 11.3.2 Cooley-Tukey FFT Algorithms

3. Features of DIF-FFT is similar to that of DIT-FFT

① 'in place' computations

② bit-reversed index

the input vector is ordered sequentially
the output vector is bit-reversed index.

③ Distance between nodes of butterfly

***N*=8 for example**

level	distance
1	4
2	2
3	1

conclusion : *m*-th level, distance is 2^{L-m}

④ Determination for W_N^r

***r* equals to the *k* left shift *m*-1 bits, and zero padded at right.**

§ 11.3.4 Inverse DFT Computation

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

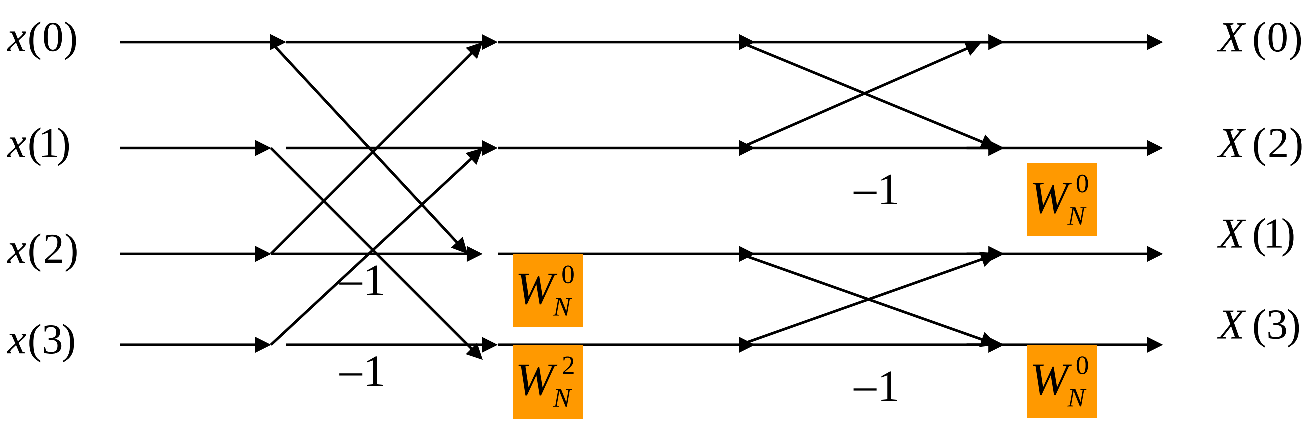
$$X(k) = DFT[x(n)] = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad 0 \le k \le N-1$$

differences:

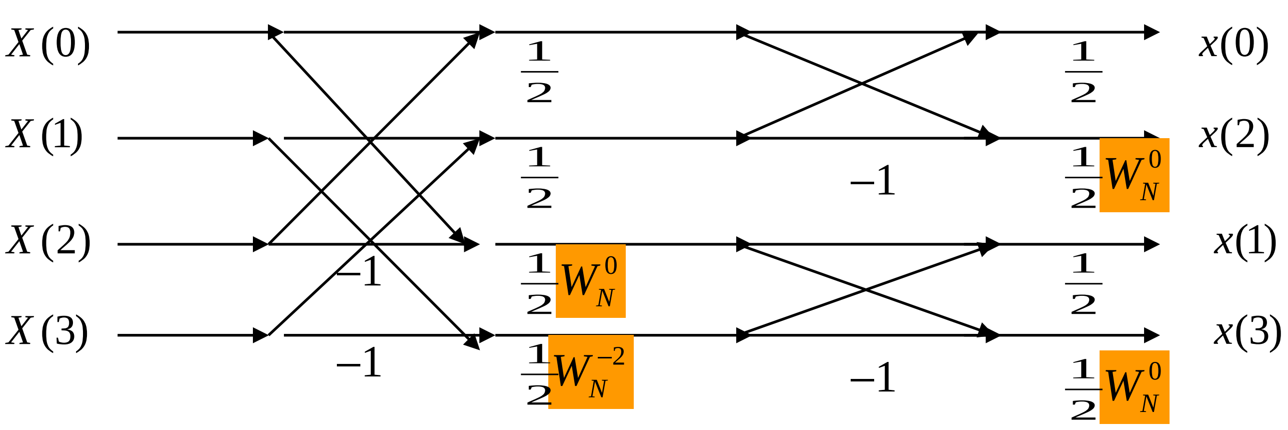
- $W_N^{nk} \quad VS \quad W_N^{-nk}$
- $coeffcent \ \frac{1}{N} \ in \ IDFT$

§ 11.3.4 Inverse DFT Computation

for example N=4, DIF-FFT



$W_N^r \rightarrow W_N^{-r}$, $x \rightarrow X$, ×1/2 each level



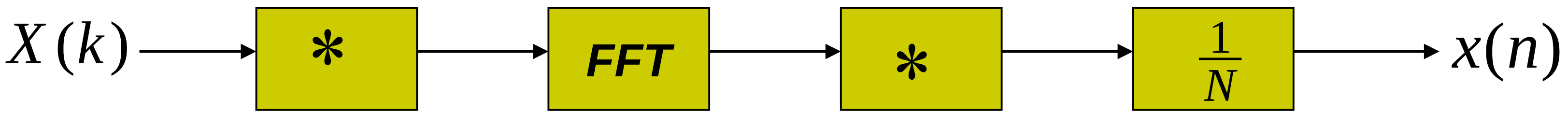
§ 11.3.4 Inverse DFT Computation

In practice

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

$$\Rightarrow \quad x(n) = \frac{1}{N}[\sum_{k=0}^{N-1} X^*(k) W_N^{nk}]^* = \frac{1}{N}\{DFT[X^*(k)]\}^*$$

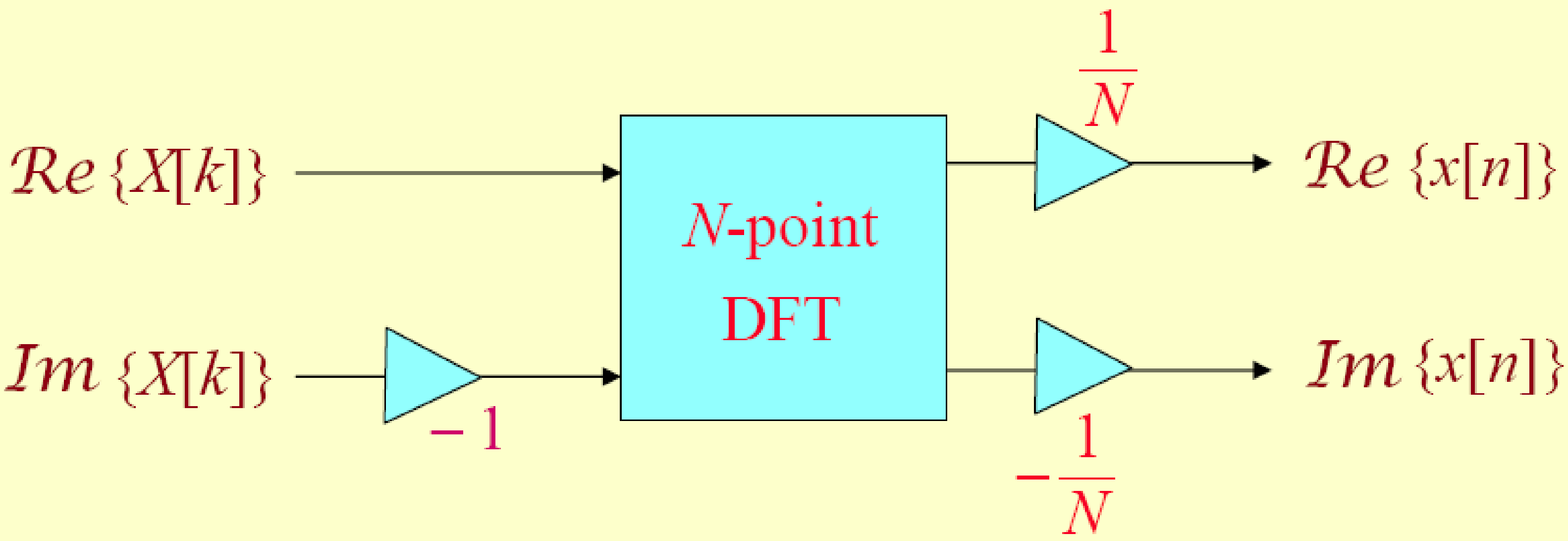
that is:



- Desired IDFT $x[n]$ is then obtained as

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

- Inverse DFT computation is shown below:



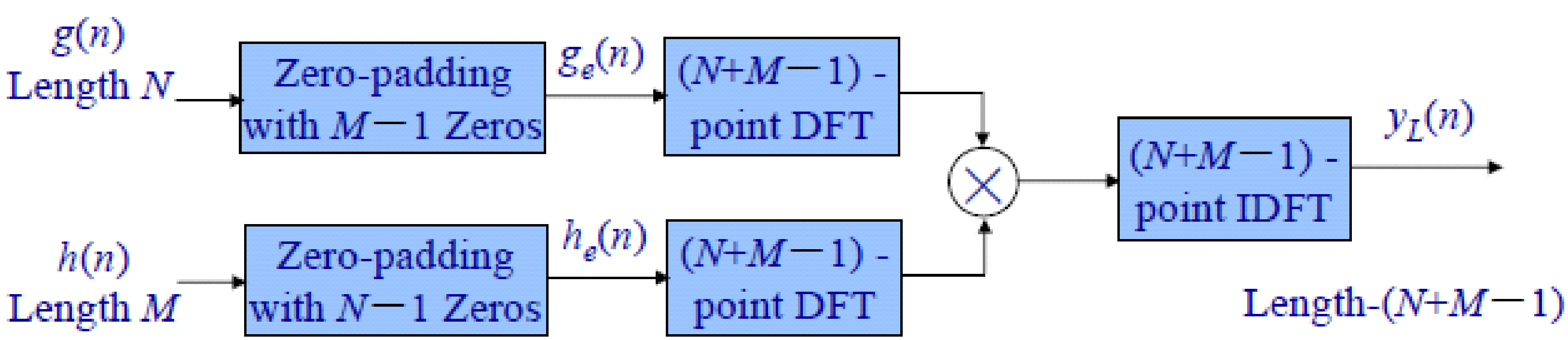
思考题：

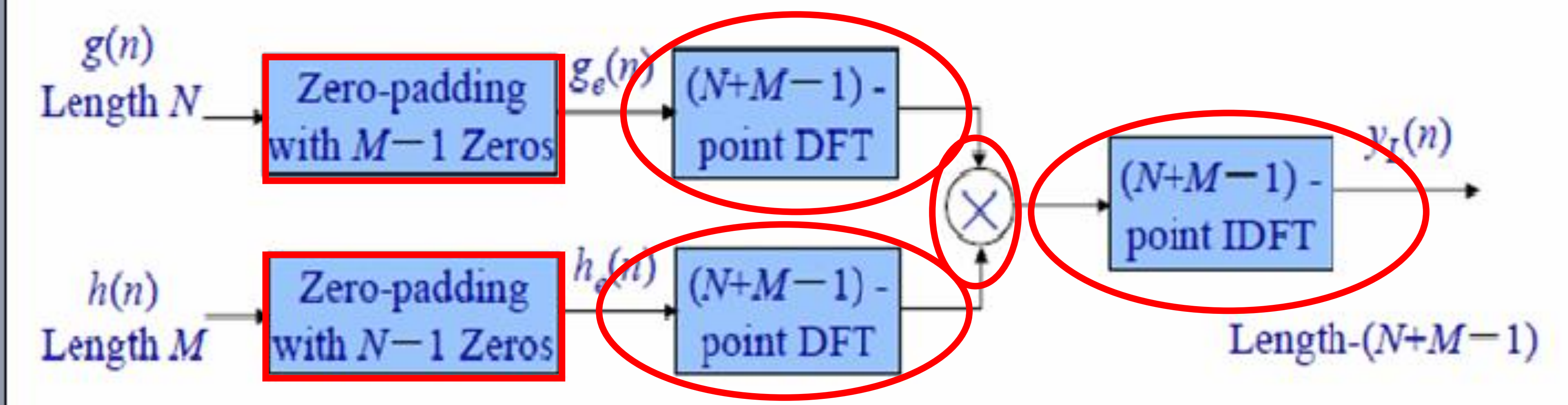
如何用快速算法实现线性卷积？

- Then

$$y_L(n)=g(n) \circledast h(n)=g(n) \, (N) \, h(n)$$

- The corresponding implementation scheme is illustrated below





- 3个L=N+M-1点的DFT，若用快速算法实现，需要 $3*L/2*\log_2 L$ 次复乘运算
- Y(k)=$G_e(k)*H_e(k)$, 两个长度为L=N+M-1的序列相乘，还需要L次复乘运算

本章重点

- 直接计算与FFT的计算量比较
- DIT-FFT，DIF-FFT算法流程
- 蝶形图画法, 蝶形运算的特点
- IDFT的快速实现
- 快速卷积的计算量分析

Homework

- **Read textbook from p.533 to 548**
- **Problem: 11.12， 11.21， 11.32**
- **Supplementary Problem:**

Using the FFT algorithm, compute the 8-point DFT of the 8-point signal $x = [4, -3, 2, 0, -1, -2, 3, 1]$.

FFT 的实现利用了 W 因子的______性和_____性？

序列{x[0], x[1],…, x[14]}进行 FFT 变换，请问复乘次数至少为______.

序列{x[0], x[1],…, x[15]}进行 DIT FFT，请写出 FFT 算法的时域输入序列：____________________.

	FFT	直接 DFT
复乘		N^2
复加		$N(N-1)$