EEE336 Signal Processing and Digital Filtering

Lecture 10 Fast Fourier Transform

10_1 Why do we introduce FFT?

Zhao Wang

Zhao.wang@xjtlu.edu.cn

Room EE322

Computational complexity

- Q: How many (complex) multiplications and additions are needed to compute DFT?

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

  - for each "*k*", we need ***N*** complex multiplications, and ***N-1*** complex additions, where W_N^{kn} **does not depend on x[n], and hence can be precomputed and saved in a table**;
  - So for *N* values of "*k*", we need $\boldsymbol{N^2}$ complex multiplications and $\boldsymbol{N(N-1)}$ complex additions;
    - The computational complexity grows with the square of the signal size.
    - This computational complexity is referred to as O(N^2), also called, order of N^2.

Properties of the Twiddle factor

- Symmetry: $\left(W_N^{kn}\right)^* = W_N^{-kn}$
- Periodicity: $W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n}$
- Reduction: $W_N^{kn} = W_{mN}^{mkn} = W_{N/m}^{kn/m}$
- Other properties:

$$W_N^{k(N-n)} = W_N^{(N-k)n}$$

$$W_N^{N/2} = -1$$

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

10_1 Wrap up

- Be able to evaluate the computational complexity!

- Be able to derive and use the properties of twiddle factors.

EEE336 Signal Processing and Digital Filtering

Lecture 10 Fast Fourier Transform

10_2 Decimation in Time: Radix-2

Zhao Wang

Zhao.wang@xjtlu.edu.cn

Room EE322

Decimation in time: radix-2

- Assume that the signal is of length $N = 2^p$, a power of two. If it is not, zero-pad the signal with enough number of zeros to ensure power-of-two length.
- N-point DFT can be computed as two N/2 point DFTs, both of which can then be computed as two N/4 point DFTs
  - Therefore an N-point DFT can be computed as four N/4 point DFTs;
  - Similarly, an N/4 point DFT can be computed as two N/8 point DFTs;
  - The entire N-point DFT can then be computed as eight N/8 point DFTs;
  - Continuing in this fashion, an N-point DFT can be computed as N/2 2-point DFTs;

DIT: Stage 1

- Stage 1: decimate the time-domain signal $x[n]$ into two half:
  - Even indexed samples: $x_0[2r]$
  - Odd indexed samples: $x_1[2r+1]$

$$\left.\begin{array}{l} x_0[2r] \\ x_1[2r+1] \end{array}\right\} r = 0, 1, \ldots, \frac{N}{2} - 1$$

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j(2\pi/N)kn} = \sum_{n \text{ even}}^{N-1} x[n] e^{-j(2\pi/N)kn} + \sum_{n \text{ odd}}^{N-1} x[n] e^{-j(2\pi/N)kn}$$

  - Substitute variables $n = 2r$ for n even and $n = 2r + 1$ for odd

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

$$= \sum_{r=0}^{N/2-1} x_0[r] W_{N/2}^{rk} + W_N^k \sum_{r=0}^{N/2-1} x_1[r] W_{N/2}^{rk}$$

$$= X_0[k] + W_N^k X_1[k], \qquad k = 0, 1, N/2-1$$

- $X_0[k]$ and $X_1[k]$ are the N/2-point DFT's of each subsequence;
- X[k] calculated here is only the first half of it, when k = 0, 1, …, N/2-1, how to get the second half of X[k]?

DIT: Stage 1

- For the second half of X[k] with k = 0, 1, …, N/2-1:

$$X_0\left[\frac{N}{2}+k\right] = \sum_{r=0}^{N/2-1} x_0[r] W_{N/2}^{(N/2+k)r} = \sum_{r=0}^{N/2-1} x_0[r] W_{N/2}^{kr} = X_0[k]$$

$$X_1\left[\frac{N}{2}+k\right] = X_1[k]$$

- So:

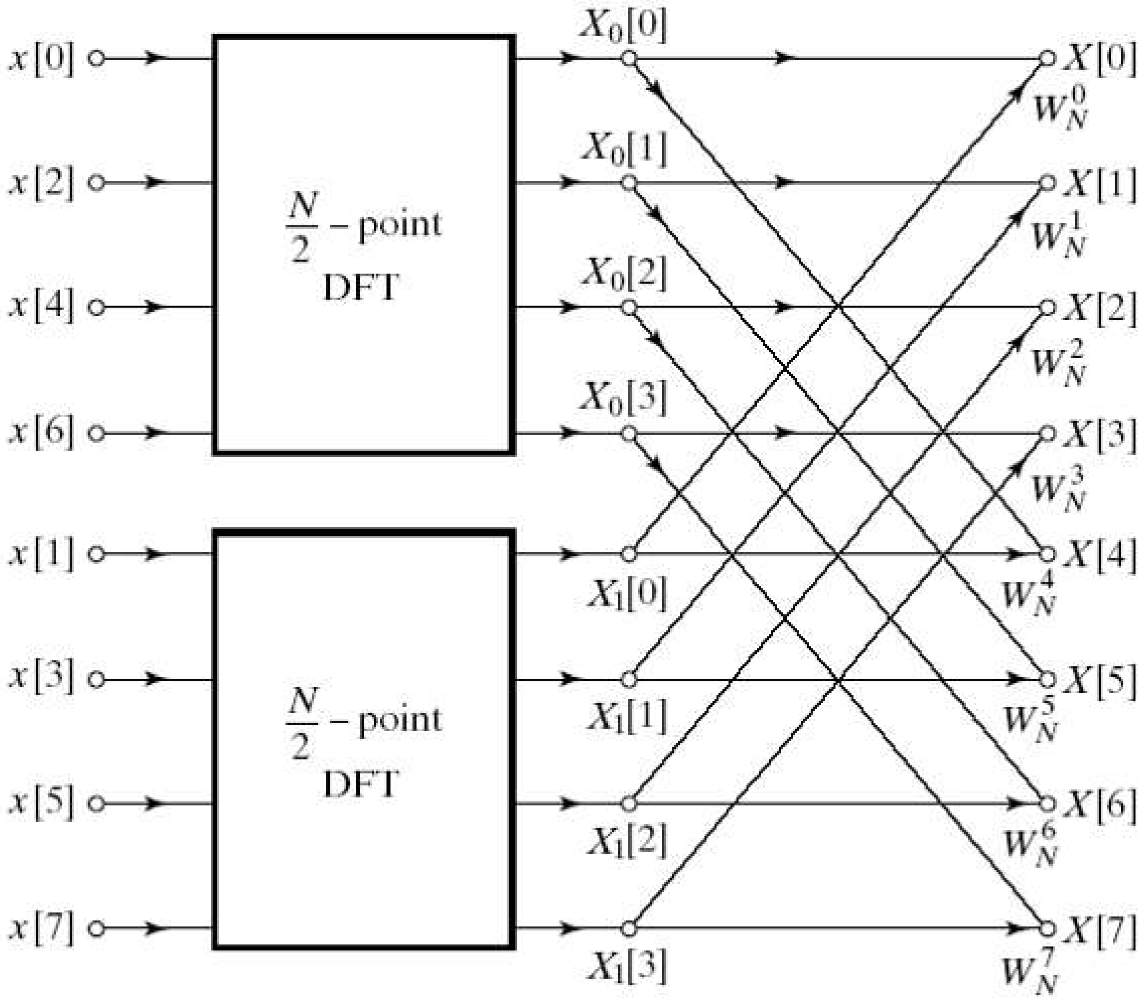
$$X\left[\frac{N}{2}+k\right] = X_0\left[\frac{N}{2}+k\right] + W_N^{N/2+k} X_1\left[\frac{N}{2}+k\right]$$

$$= X_0[k] + W_N^{k+N/2} X_1[k]$$

$$= X_0[k] - W_N^{k} X_1[k]$$

DIT: Stage 1

- 8-point DFT example using decimation-in-time
- Two N/2-point DFTs
  - $2(N/2)^2$ complex multiplications
  - $2(N/2)^2$ complex additions
- Combining the DFT outputs
  - N complex multiplications
  - N complex additions
- Total complexity
  - $N^2/2+N$ complex multiplications
  - $N^2/2+N$ complex additions
  - More efficient than direct DFT
- No the end!



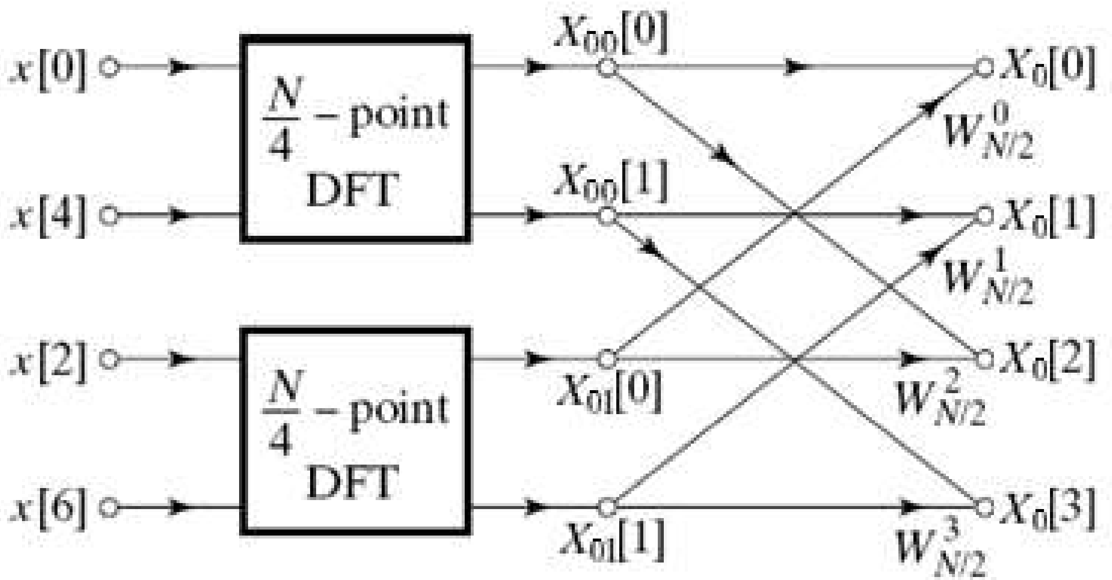
DIT: Stage 2

- Repeat same process
  - Divide N/2-point DFTs into two N/4-point DFTs

$$X_0[k] = X_{00}[\langle k \rangle_{N/4}] + W_{N/2}^{k} X_{01}[\langle k \rangle_{N/4}], \qquad 0 \le k \le \frac{N}{4} - 1$$

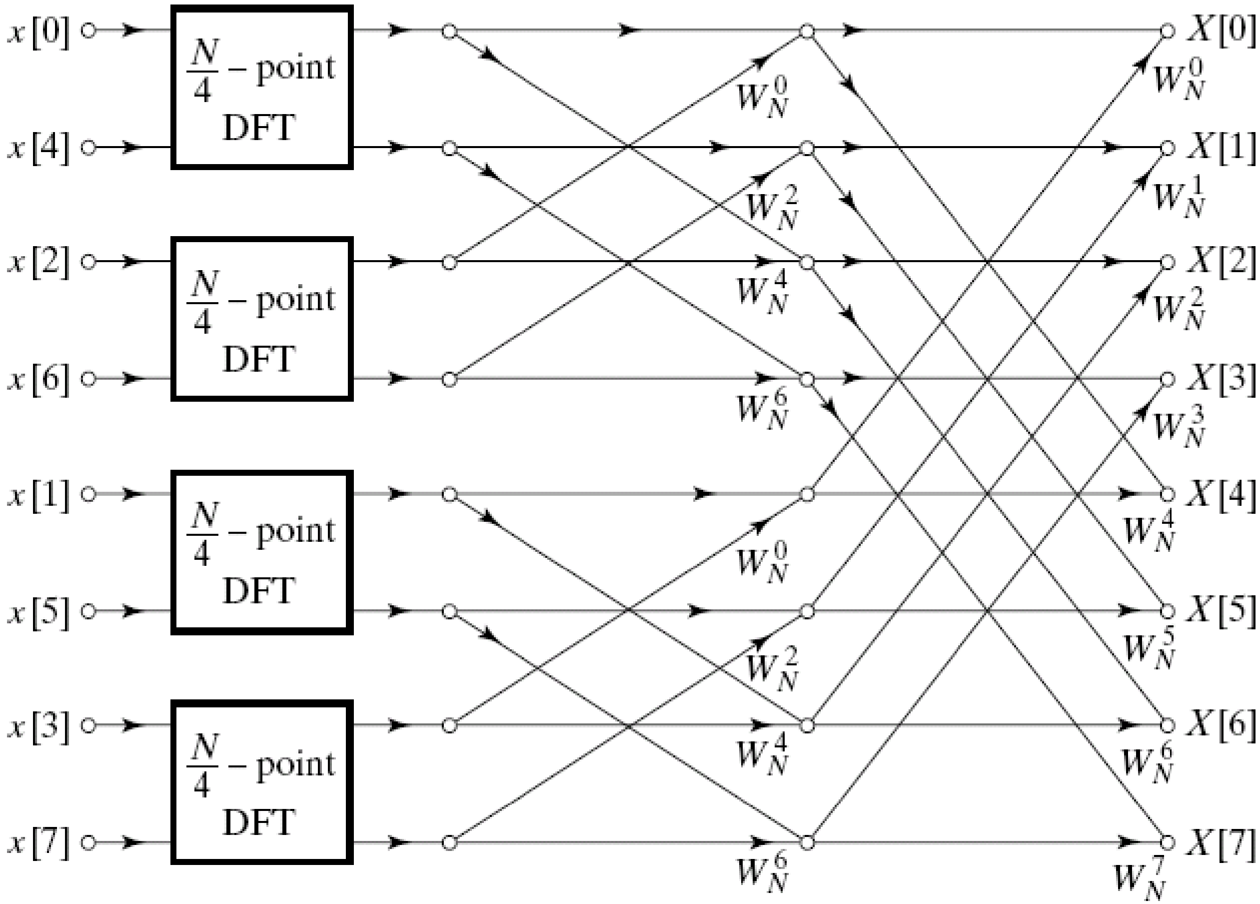
$$X_0[k] = X_{00}[\langle k \rangle_{N/4}] + W_{N/2}^{k+N/2} X_{01}[\langle k \rangle_{N/4}], \qquad \frac{N}{4} \le k \le \frac{N}{2} - 1$$

  - Combine outputs



DIT cont.

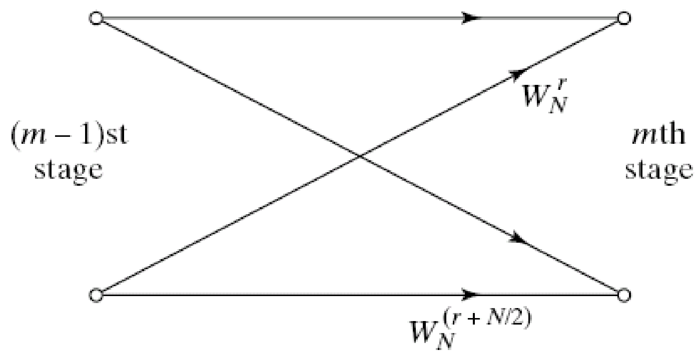
- After two stages of decimation:



- Repeat until we're left with two-point DFT's

Butterfly

- Flow graph constitutes of butterflies

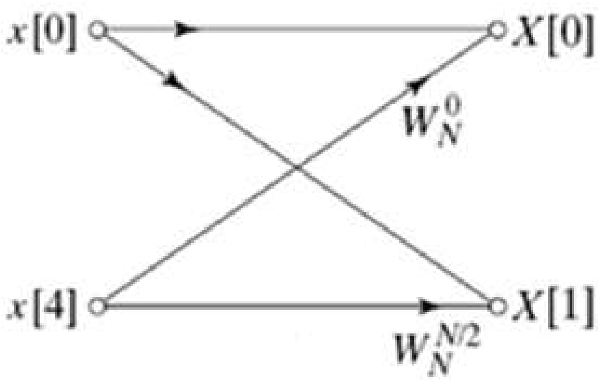


Two complex multiplication
Two complex addition

- How many operations do we need for 2-point DFT?

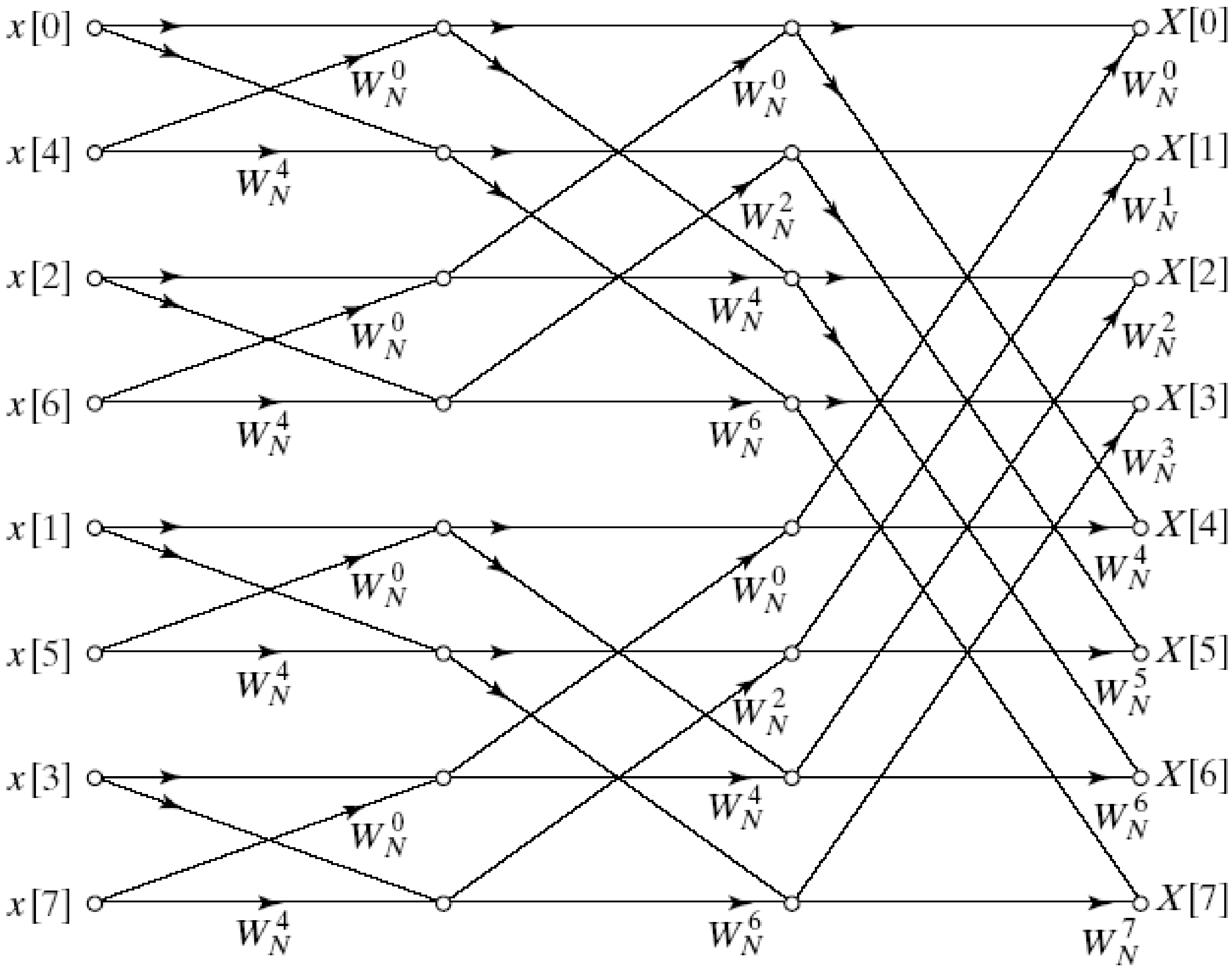
$$X[0] = x[0] + W_N^0 x[1]$$
$$X[1] = x[0] + W_N^4 x[1] \;= x[0] - x[1]$$

- The butterfly:



No complex multiplication
Two complex addition

Final flow-graph of DIT-2



- Number of stages:

$$p = log_2 N$$

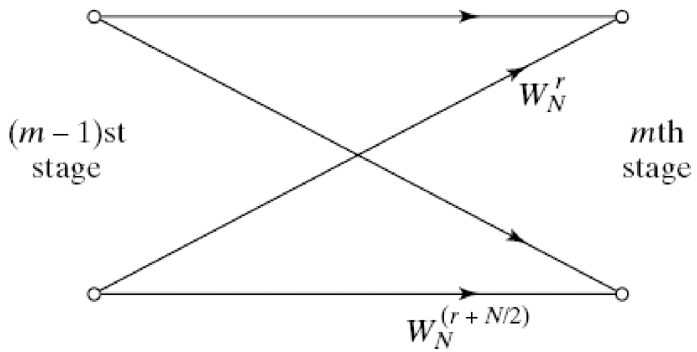
- Number of butterflies per stage:

$$N/2$$

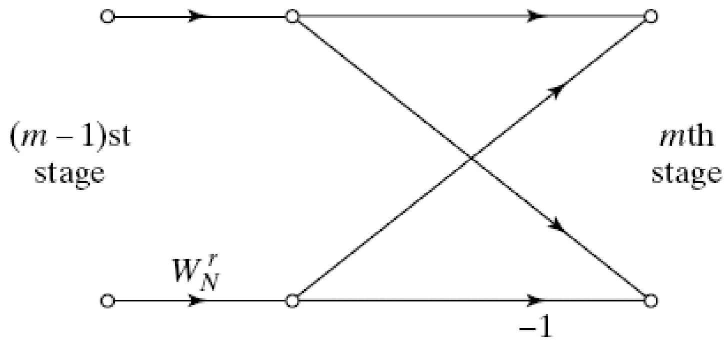
- Two computational complexity:
  - $N(p\text{-}1) = N(\log_2 N\text{-}1)$ complex multiplications;
  - $Np = N\log_2 N$ complex additions.

Simplify the butterfly process

- Original butterfly:



- We can implement each butterfly with one multiplication



- Final complexity for decimation-in-time FFT
  - (N/2)($\log_2$N-1) complex multiplications and N$\log_2$N additions

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Comparison of computational complexity

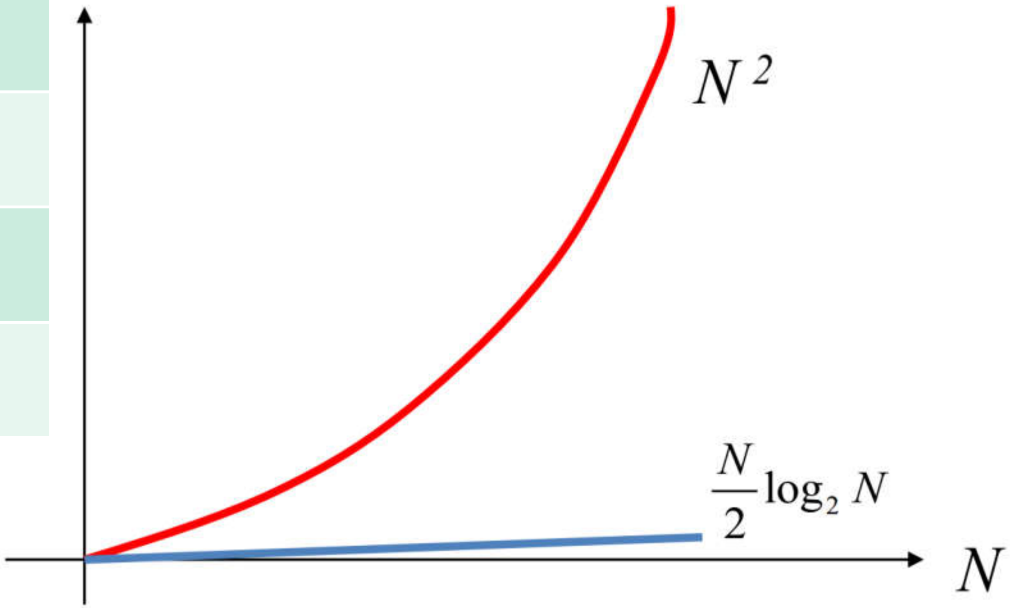
Length (N)	DFT		FFT	
	No. of *	No. of +	No. of *	No. of +
4	16	12	2	8
8	64	56	8	24
16	256	240	24	64
32	1024	992	64	160
64	4096	4032	160	384
128	16384	16256	384	896
256	65536	65280	896	2048

For DFT

- No. of * is N^2
- No. of + is $N(N-1)$

For FFT

- No. of * is $\frac{N}{2}(\log_2 N - 1)$
- No. of + is $N \log_2 N$



Bit-reversal

- Note the arrangement of the input indices - Bit reversed indexing

x[n]
x[0]
x[4]
x[2]
x[6]
x[1]
x[5]
x[3]
x[7]

X[k]
X[0]
X[1]
X[2]
X[3]
X[4]
X[5]
X[6]
X[7]

$x[n_2]$
x[000]
x[100]
x[010]
x[110]
x[001]
x[101]
x[011]
x[111]

$X[k_2]$
X[000]
X[001]
X[010]
X[011]
X[100]
X[101]
X[110]
X[111]

10_2 Wrap up

- Understand how the FFT is performed
- Be able to draw the flow graph of FFT-DIT-2
- Be able to calculate any value in the graph
- Be able to calculate the computational complexity of different length of N
- Be able to arrange the input x[n] in correct order according to bit-reversal scheme

EEE336 Signal Processing and Digital Filtering

Lecture 10 Fast Fourier Transform

10_3 Applications of FFT

(Stream data processing)

Zhao Wang

Zhao.wang@xjtlu.edu.cn

Room EE322

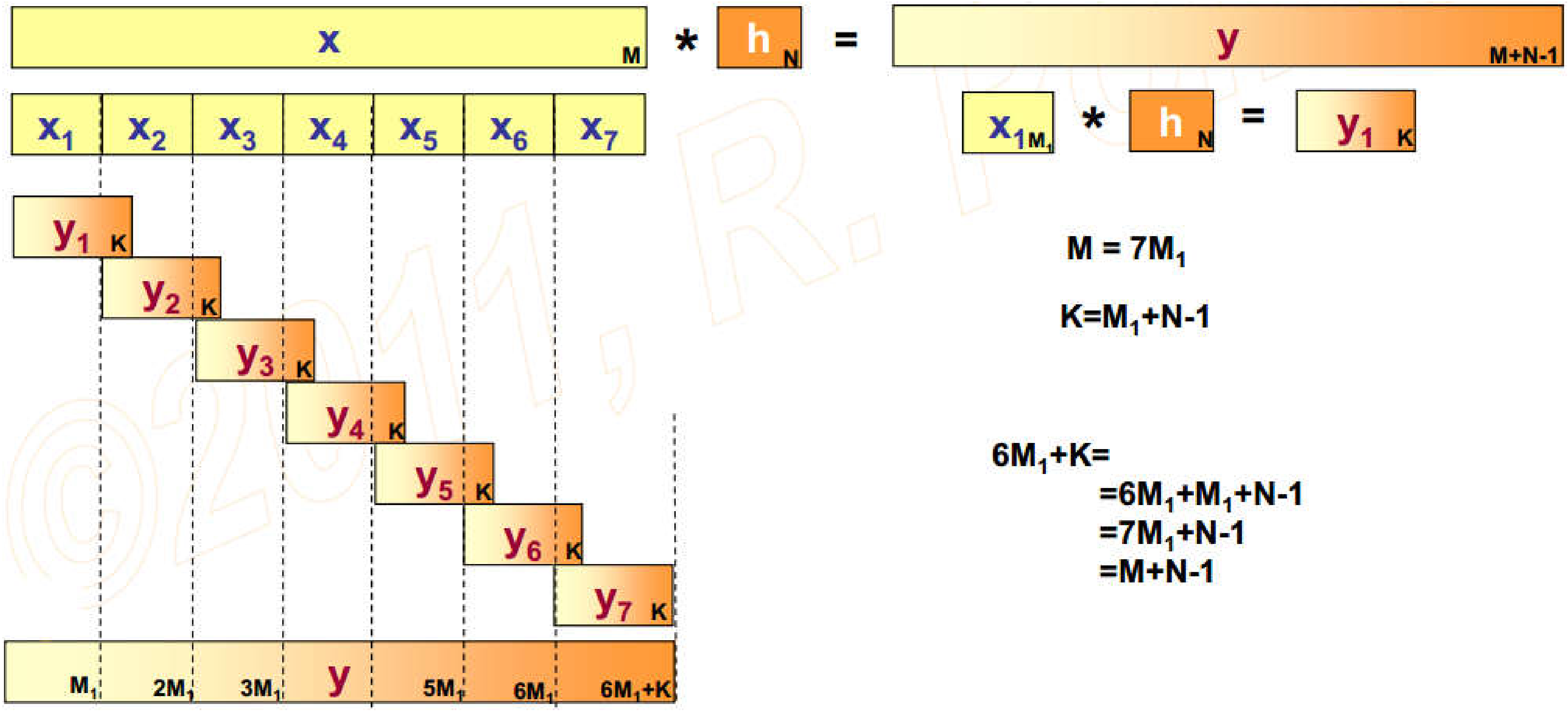
Filtering Streaming Data

- In most real-world applications, the filter length is actually rather small (typically, N<100); however, the input signal is obtained as a streaming data, and therefore can be very long.
- To calculate the output of the filter, we can:
  - a) Wait until we receive all the data, and then do a full linear convolution of length N filter and length M input x[n], where M>>>>N -> **y[n]=x[n]*h[n]**
  - b) We can use the DFT based method -> **y[n]=IDFT(X[k].H[k])**, but we still need to wait for the entire data to arrive to calculate **X[k]**
- In either case, the calculation is very long, expensive, and need to wait for the entire data to arrive.

- Can we just process the data in batches, say 1000 samples at a time, and then concatenate the results…?

Overlap Add

- The border effect! Processing individual segments separately and then combining them is possible, however, the border distortion needs to be addressed
  - Overlap add is a method that allows us to compute the individual segments and then concatenate them in such a way that the concatenated signal is the same as the one that would be obtained if we processed the entire data at once.



Overlap Add

- We fist segment x[n], assumed to be a causal sequence here without any loss of generality, into a set of continuous finite-length subsequences $x_m[n]$ of length M_1 each:

$$x[n] = \sum_{m=0}^{\infty} x_m[n - mM_1], \qquad where\ x_m[n] = \begin{cases} x[n + mM_1], 0 \le n \le M_1 - 1 \\ 0, \qquad\qquad otherwise \end{cases}$$

- Thus we can write

$$y[n] = h[n] * x[n] = \sum_{m=0}^{\infty} y_m[n - mM_1], \qquad where\ y_m[n] = h[n] * x_m[n]$$

  - Since h[n] is of length N and $x_m[n]$ is of length M_1, the liner convolution $h[n]*x_m[n]$ is of length $N+M_1-1$;
  - As a result, the desired linear convolution y[n] has been broken up into a sum of infinite number of short-length linear convolutions y_m of length $N+M_1-1$;
  - Each of these short convolutions can be implemented using the zero-padding based DFT method, which are computed on the basis of $N+M_1-1$ points.

Overlap Add

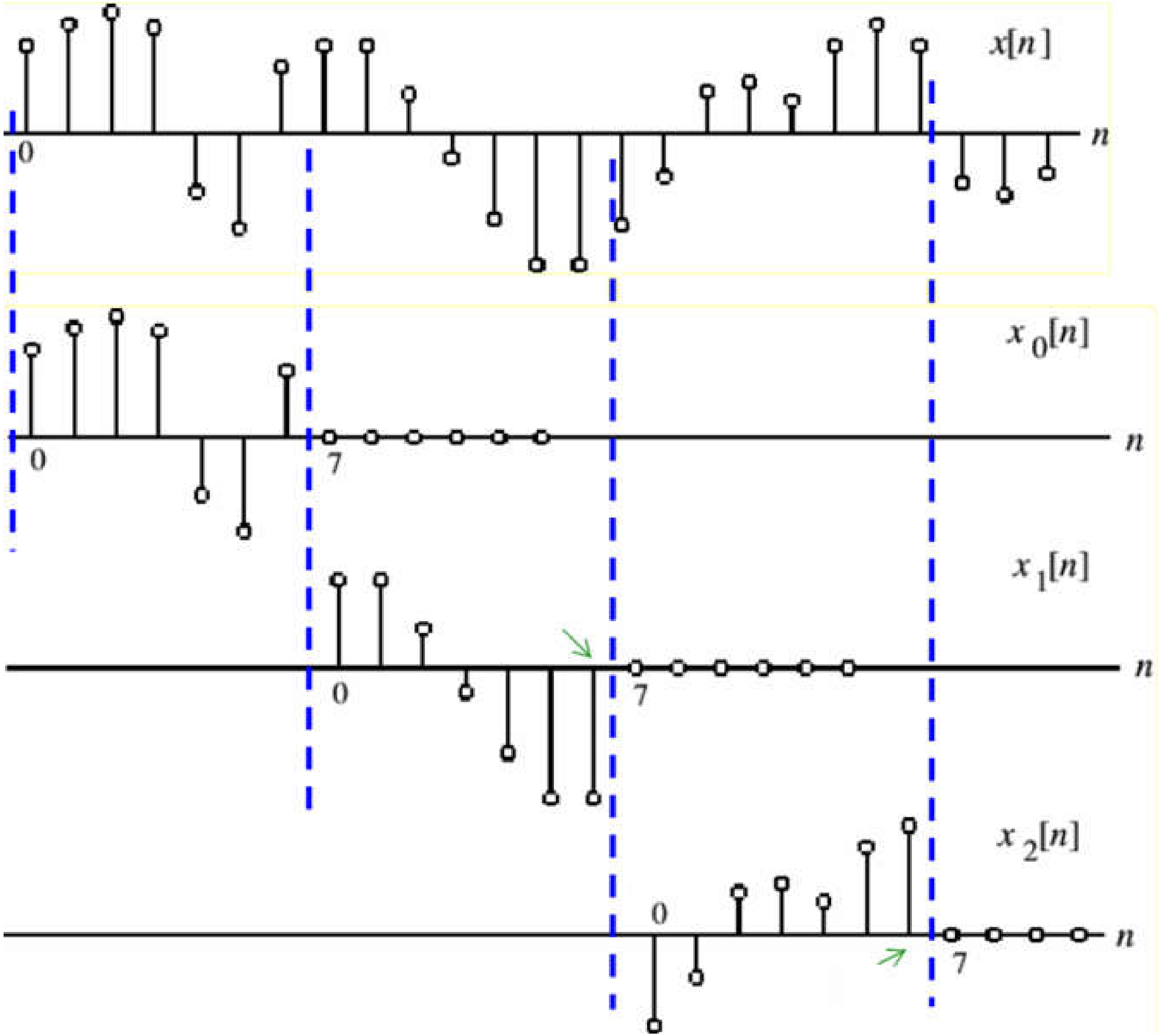
- Notice
  - The first convolution y1 is of length $N + M_1 - 1$ and is defined on $0 \leq n \leq N + M_1 - 2$;
  - The second convolution y2 is also of length $N + M_1 - 1$, but is defined on $M_1 \leq n \leq 2M_1 + N - 2$;
  - There is an overlap of N-1 samples between these two short linear convolutions.

- In general, there will be an overlap of N-1 samples between all the adjacent results of the short convolutions.

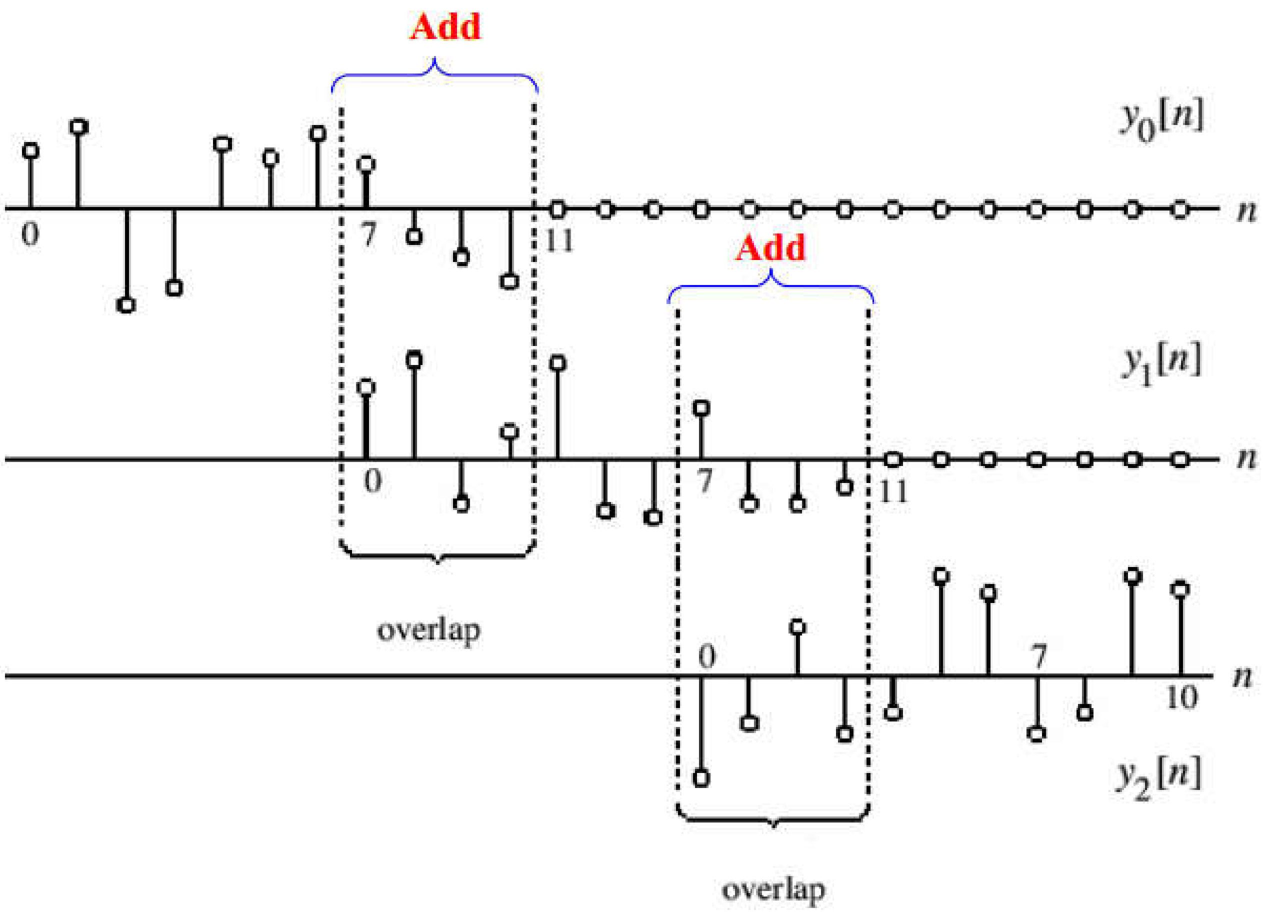
Overlap Add

$N = 5$

$M_1 = 7$



Overlap Add



- Therefore, y[n] should be given by:

$$y[n] = y_0[n], 0 \leq n \leq 6$$

$$y[n] = y_0[n] + y_1[n-7], 7 \leq n \leq 10$$

$$y[n] = y_1[n-7], 11 \leq n \leq 13$$

$$y[n] = y_1[n-7] + y_2[n-14], 14 \leq n \leq 17$$

10_3 Wrap up

- Real-time signal processing:
  - Long (or infinite) input data x[n];
  - Short system impulse h[n];

- Output y[n] can be obtained by segment processing:
  - Overlap-add
    - How to perform
    - Evaluate the computational complexity

Chapter 10 Summary

- Computational complexity
  - Convolution, DFT, FFT
- FFT: DIT-2
  - Twiddle factor properties
  - DIT process, flow graph, gain on each path, etc.
  - Bit reversal
- Application: streaming data
  - Overlap add method