18-491/691 Lecture #16
FAST FOURIER TRANSFORM ALTERNATE IMPLEMENTATIONS

Richard M. Stern

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Phone: +1 (412) 268-2535
FAX: +1 (412) 268-3890
rms@cs.cmu.edu
http://www.ece.cmu.edu/~rms

March 24, 2021

Introduction

- **In our lecture last Monday we described and discussed the basic decimation-in-time Cooley-Tuckey fast Fourier transform algorithm for DFT sizes that are integer powers of 2 (radix 2)**
- **Today we will discuss some variations and extensions of the basic FFT algorithm:**
  - Computation of the inverse FFT
  - One further trivial efficiency
  - Alternate forms of the FFT structure
  - The decimation-in-frequency FFT algorithm
  - FFT structures for DFT sizes that are not an integer power of 2

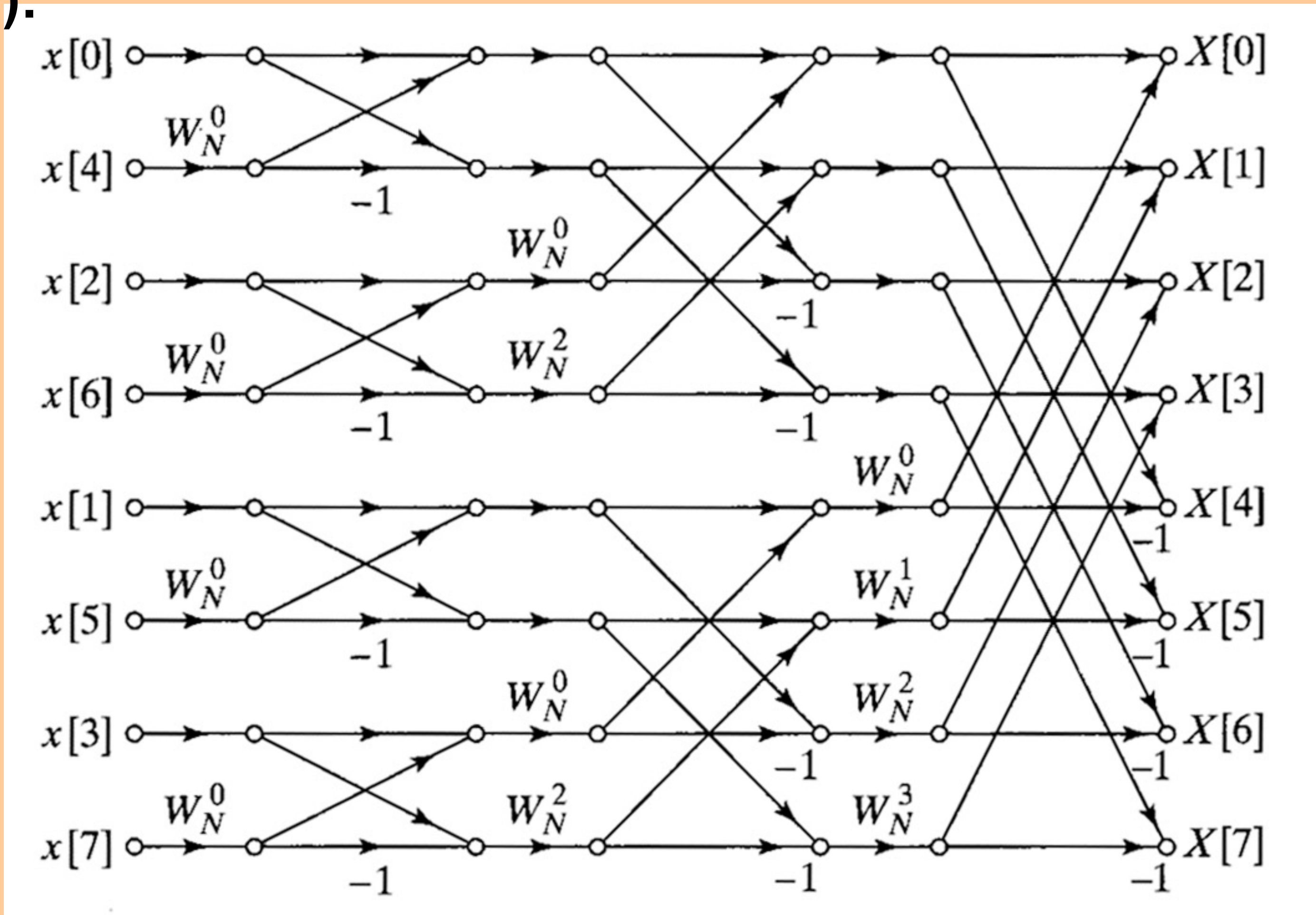
Alternate FFT structures

- **We developed the basic decimation-in-time (DIT) FFT structure in the last lecture, but other forms are possible simply by rearranging the branches of the signal flowgraph**

- **Some issues to consider:**
  - Natural or bit-reversed input and output?
  - In-place computation?

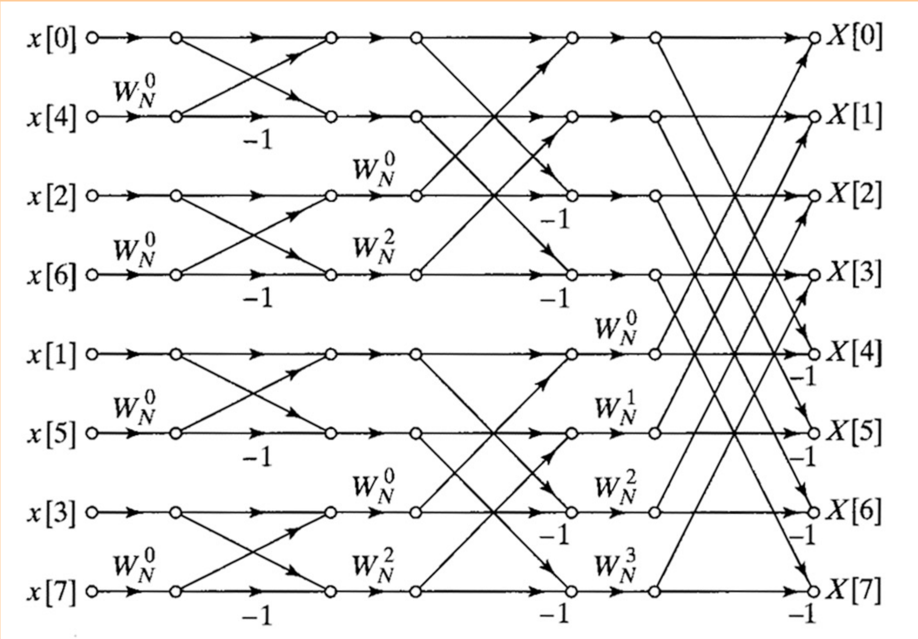
Alternate DIT FFT structures (continued)

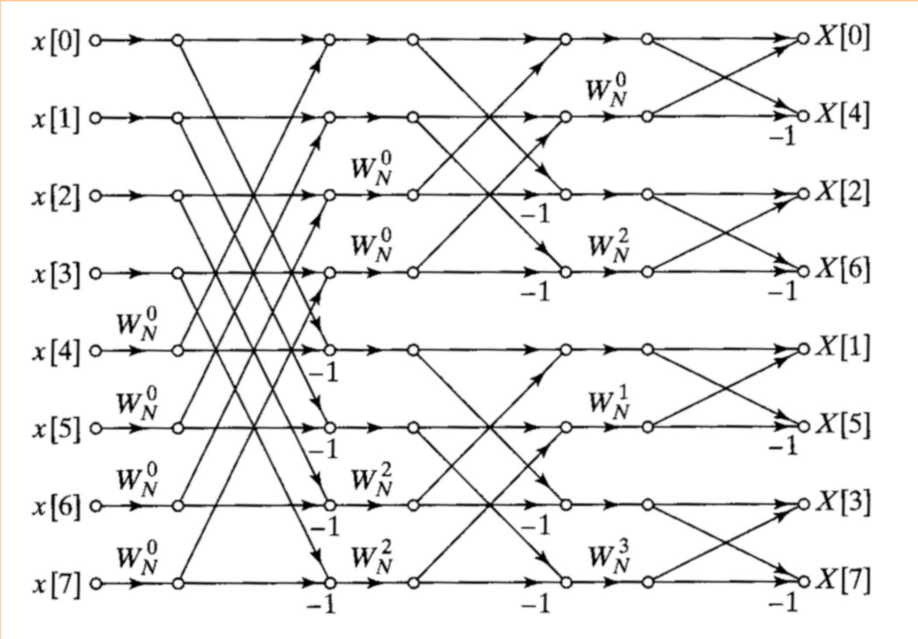
- **DIT structure with input bit-reversed, output natural (OSYP 9.11):**



Alternate DIT FFT structures (continued)

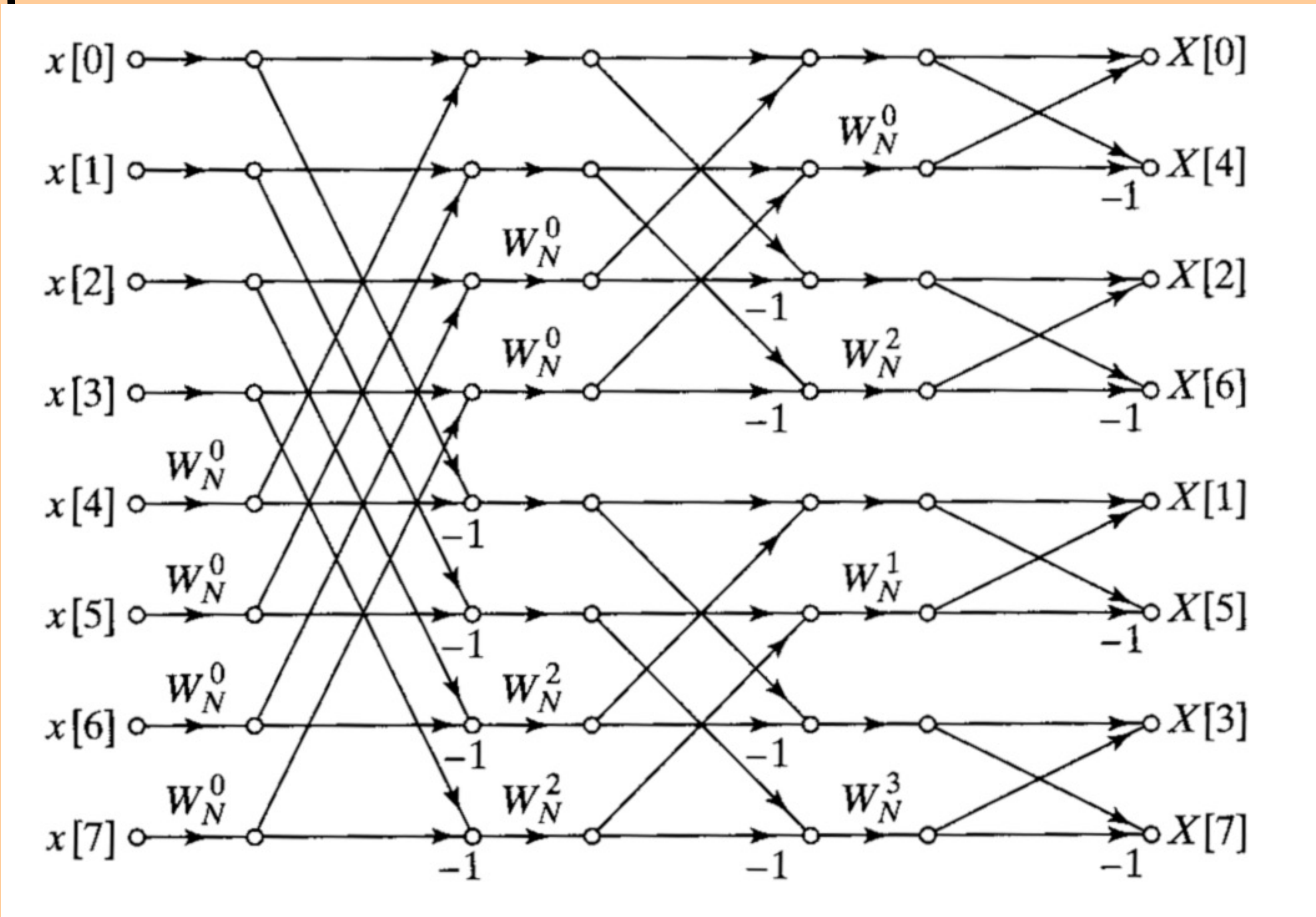
- **The original DIT structure (OSYP 9.11):**
- **Rearranged structure (OSYP 9.15):**





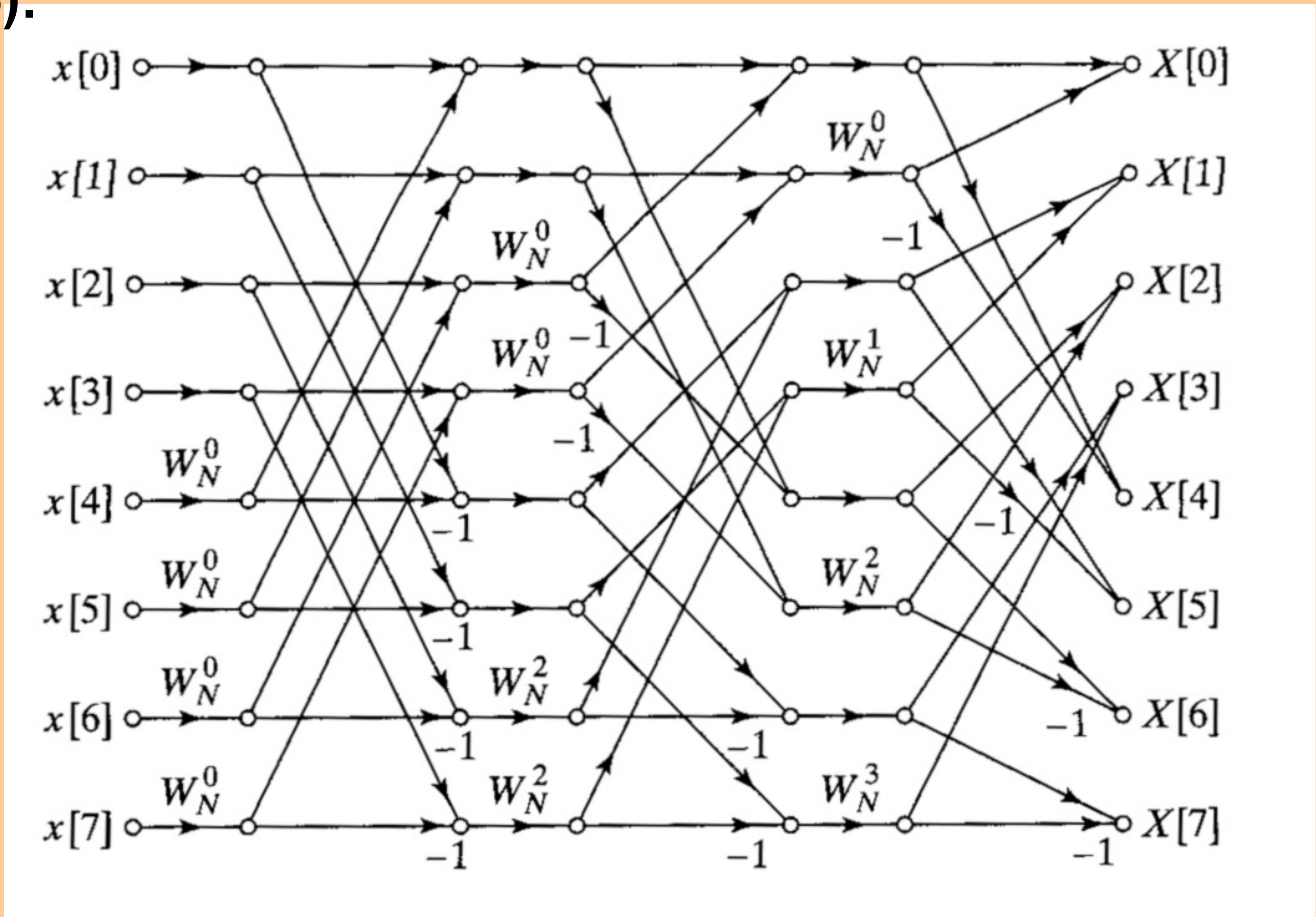
Alternate DIT FFT structures (continued)

- **DIT structure with input natural, output bit-reversed (OSYP 9.15):**



Alternate DIT FFT structures (continued)

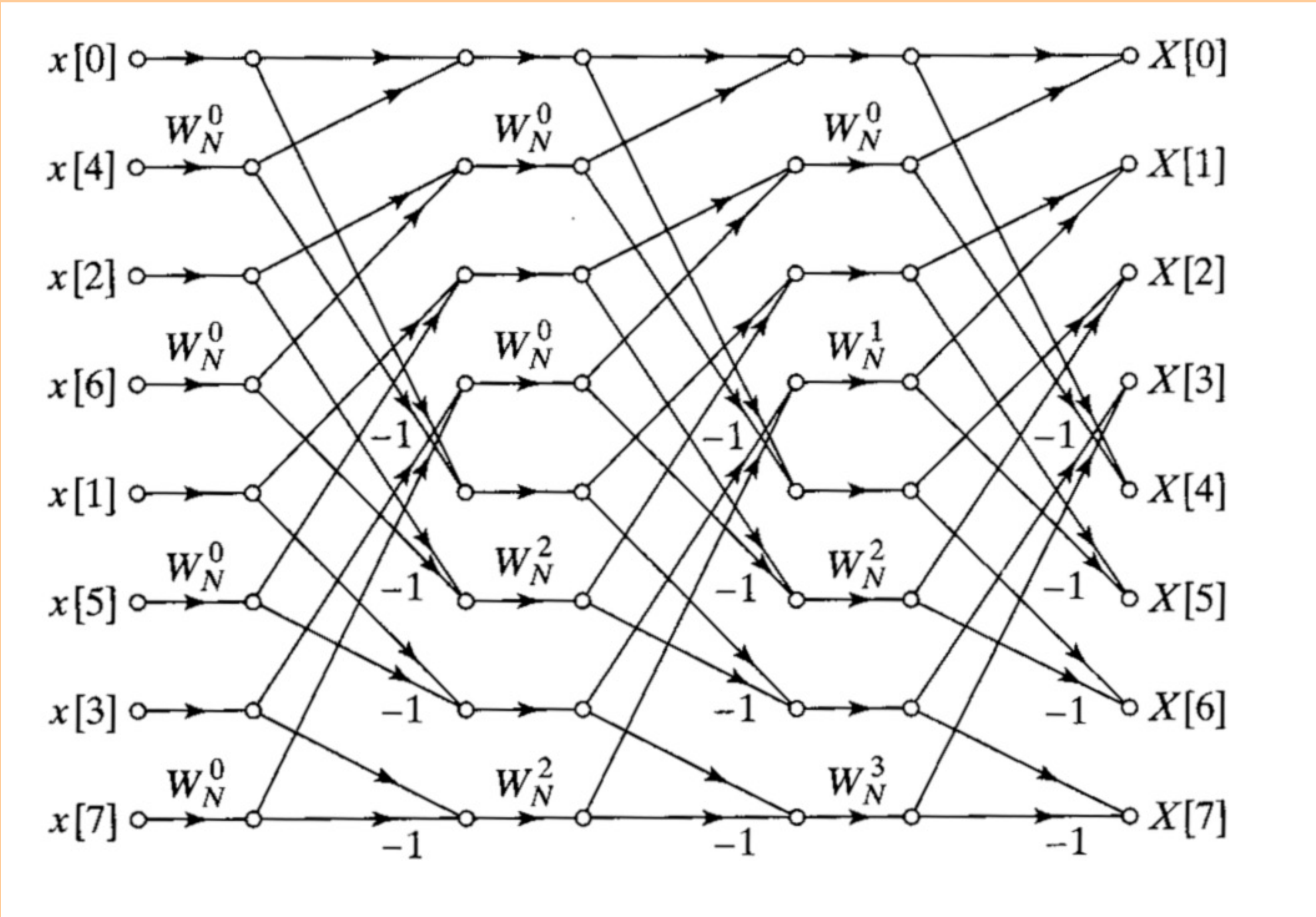
- **DIT structure with both input and output in natural order (OSYP 9.16):**



Carnegie Mellon

Alternate DIT FFT structures (continued)

- DIT structure with same structure for each stage (OSYP 9.17):



Comments on alternate FFT structures

- **A method to avoid bit-reversal in filtering operations is:**
  - Compute forward transform using natural input, bit-reversed output (as in OSB 9.10)
  - Multiply DFT coefficients of input and filter response (both in bit-reversed order)
  - Compute inverse transform of product using bit-reversed input and natural output (as in OSB 9/14)
- **Latter two topologies (as in OSYP 9.16 and 9.17) are now rarely used**

The decimation-in-frequency (DIF) FFT algorithm

- Introduction: Decimation in frequency is an alternate way of developing the FFT algorithm
- It is different from decimation in time in its development, although it leads to a very similar structure

The decimation in frequency FFT (continued)

- Consider the original DFT equation ….

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

- Separate the first half and the second half of time samples:

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

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

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

- Note that these are not *N/2*-point DFTs

Continuing with decimation in frequency ...

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

- **For *k* even, let** $k = 2r$

$$X[k] = \sum_{n=0}^{(N/2)-1} \left[x[n] + (-1)^{2r} x[n+(N/2)] \right] W_N^{n2r} = \sum_{n=0}^{(N/2)-1} [x[n] + x[n+(N/2)]] W_{N/2}^{nr}$$

- **For *k* odd, let** $k = 2r+1$

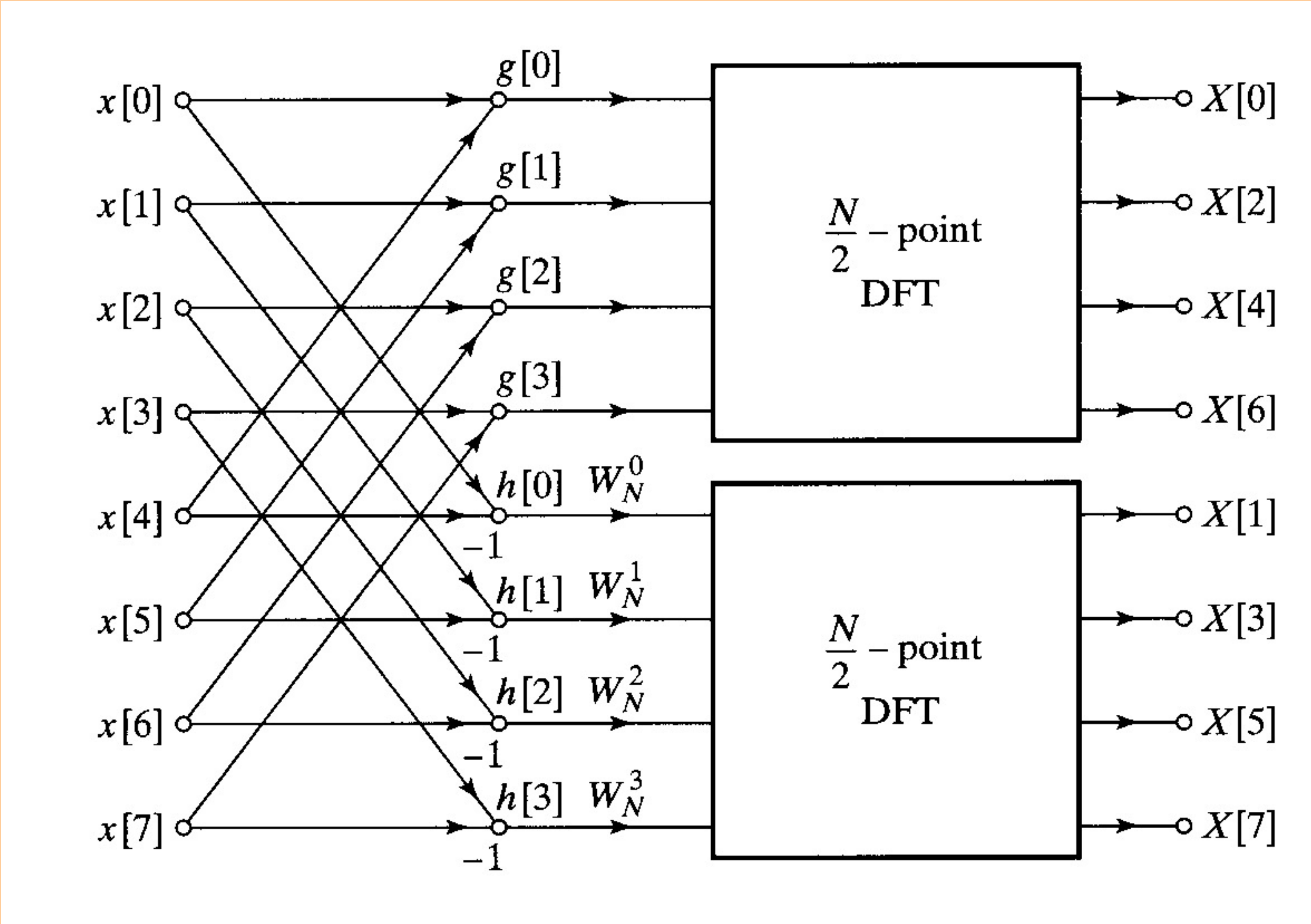
$$X[k] = \sum_{n=0}^{(N/2)-1} \left[x[n] + (-1)^{2r}(-1) x[n+(N/2)] \right] W_N^{n(2r+1)}$$

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

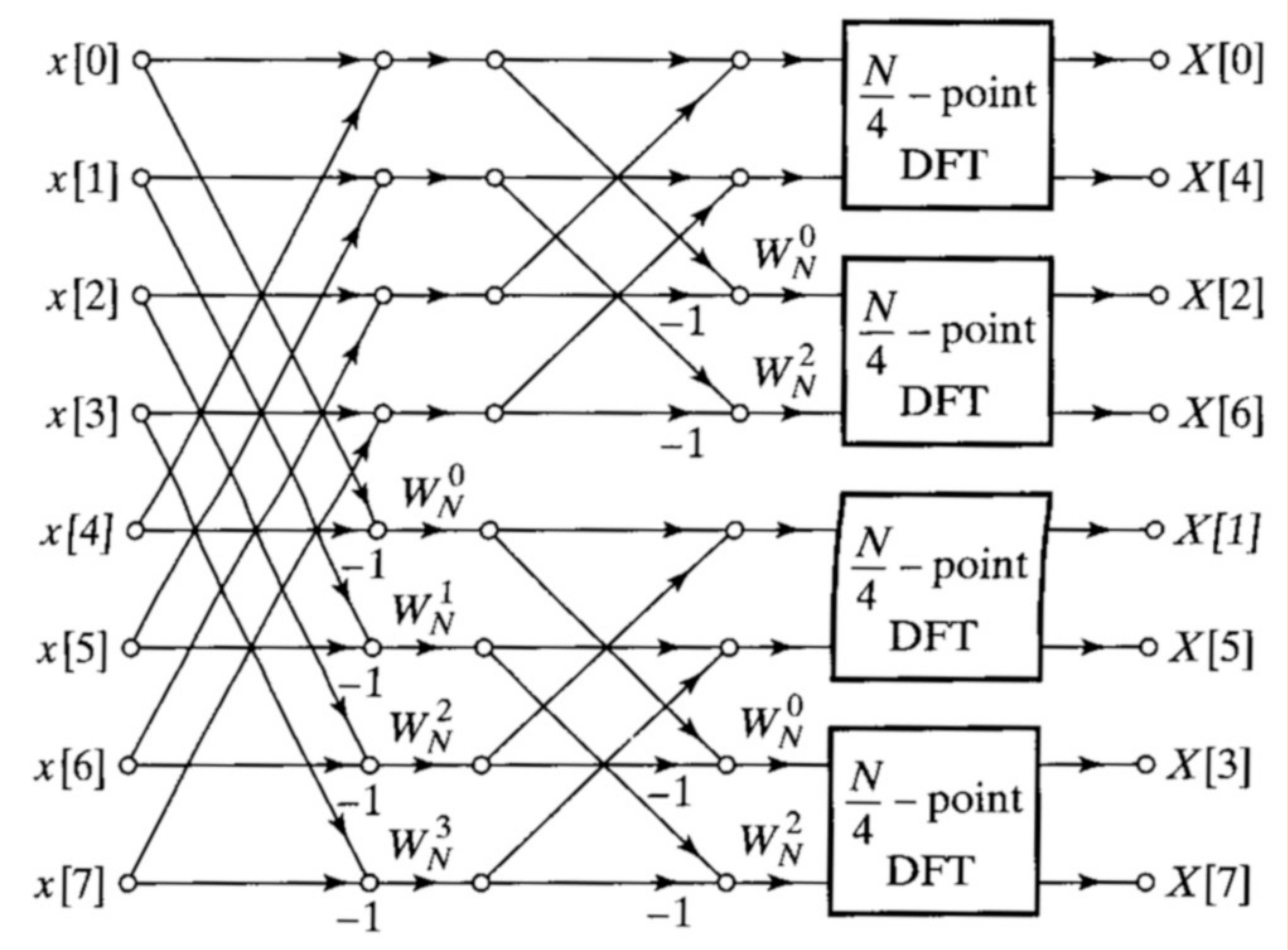
- **These expressions are the *N/2*-point DFTs of**

$$x[n] + x[n+(N/2)] \text{ and } [x[n] - x[n+(N/2)]] W_N^n$$

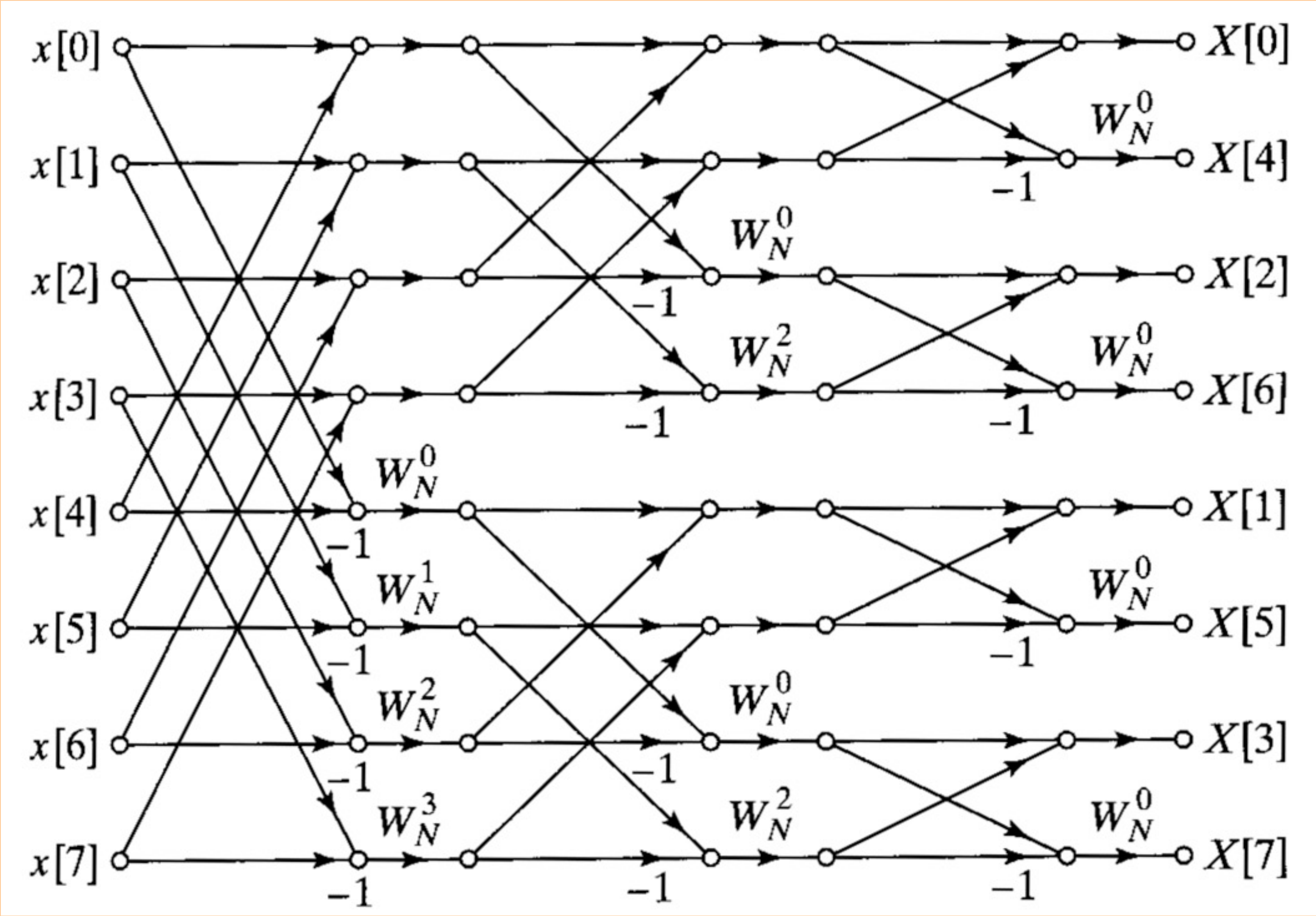
These equations describe the following structure:



Continuing by decomposing the odd and even *output* points we obtain …

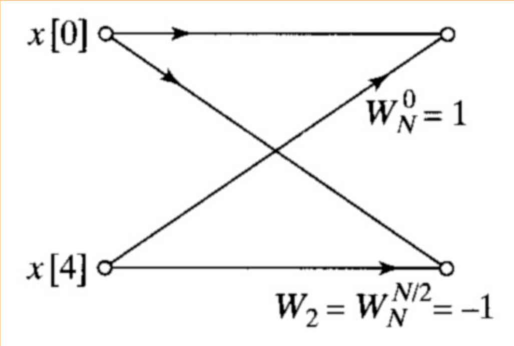


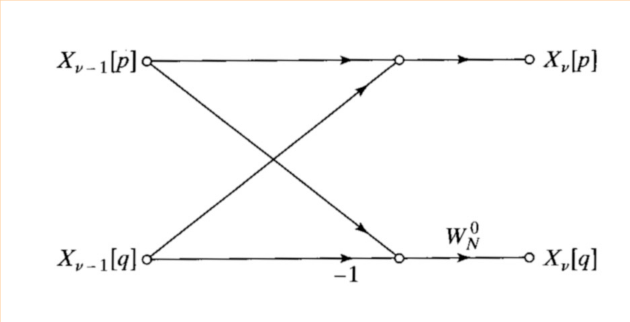
… and replacing the *N/4*-point DFTs by butterflys we obtain



The DIF FFT is the *transpose* of the DIT FFT

- **To obtain flowgraph transposes:**
  - Reverse direction of flowgraph arrows
  - Interchange input(s) and output(s)
- **DIT butterfly:** **DIF butterfly:**



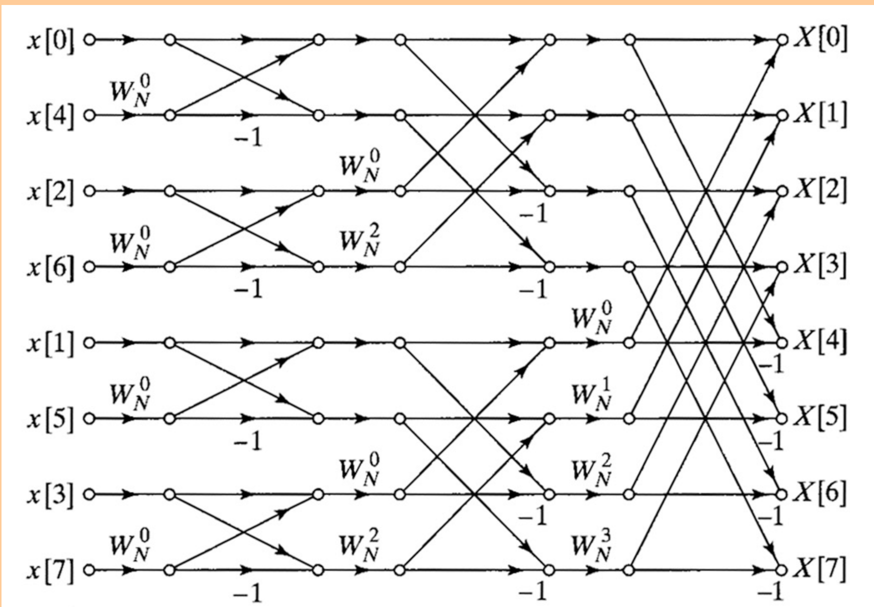


- **Comment:**
  - We will revisit transposed forms again in our discussion of filter implementation

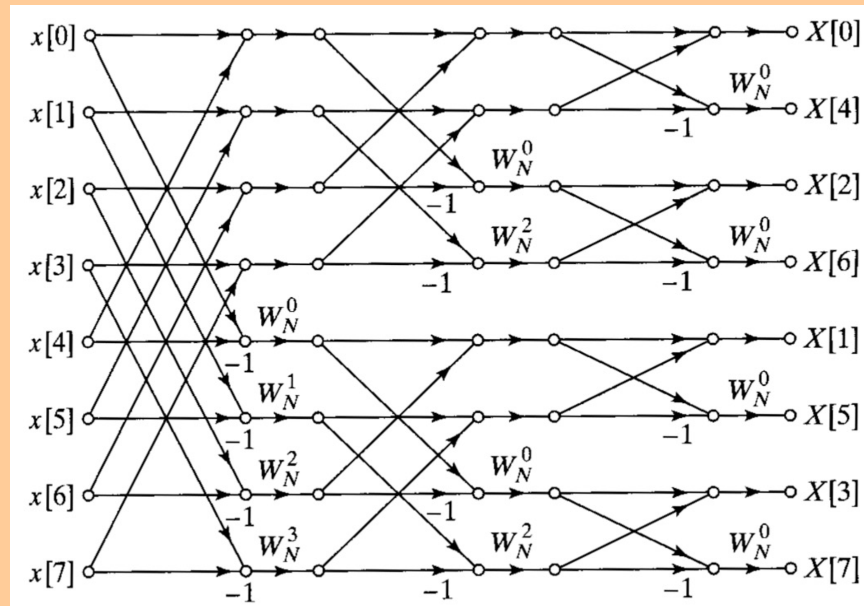
The DIF FFT is the *transpose* of the DIT FFT

- **Comparing DIT and DIF structures:**

DIT FFT structure:



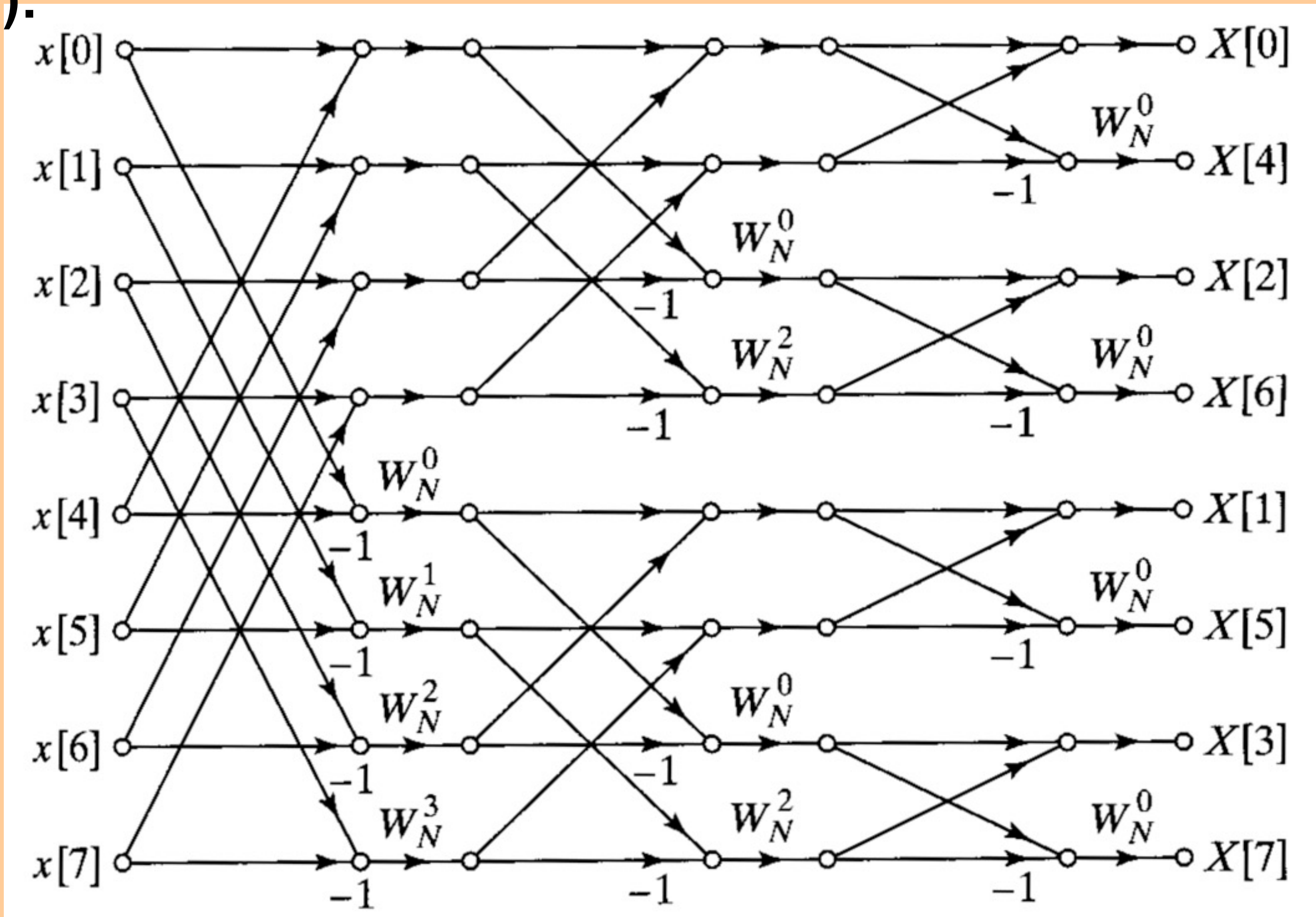
DIF FFT structure:



- **Alternate forms for DIF FFTs are similar to those of DIT FFTs**

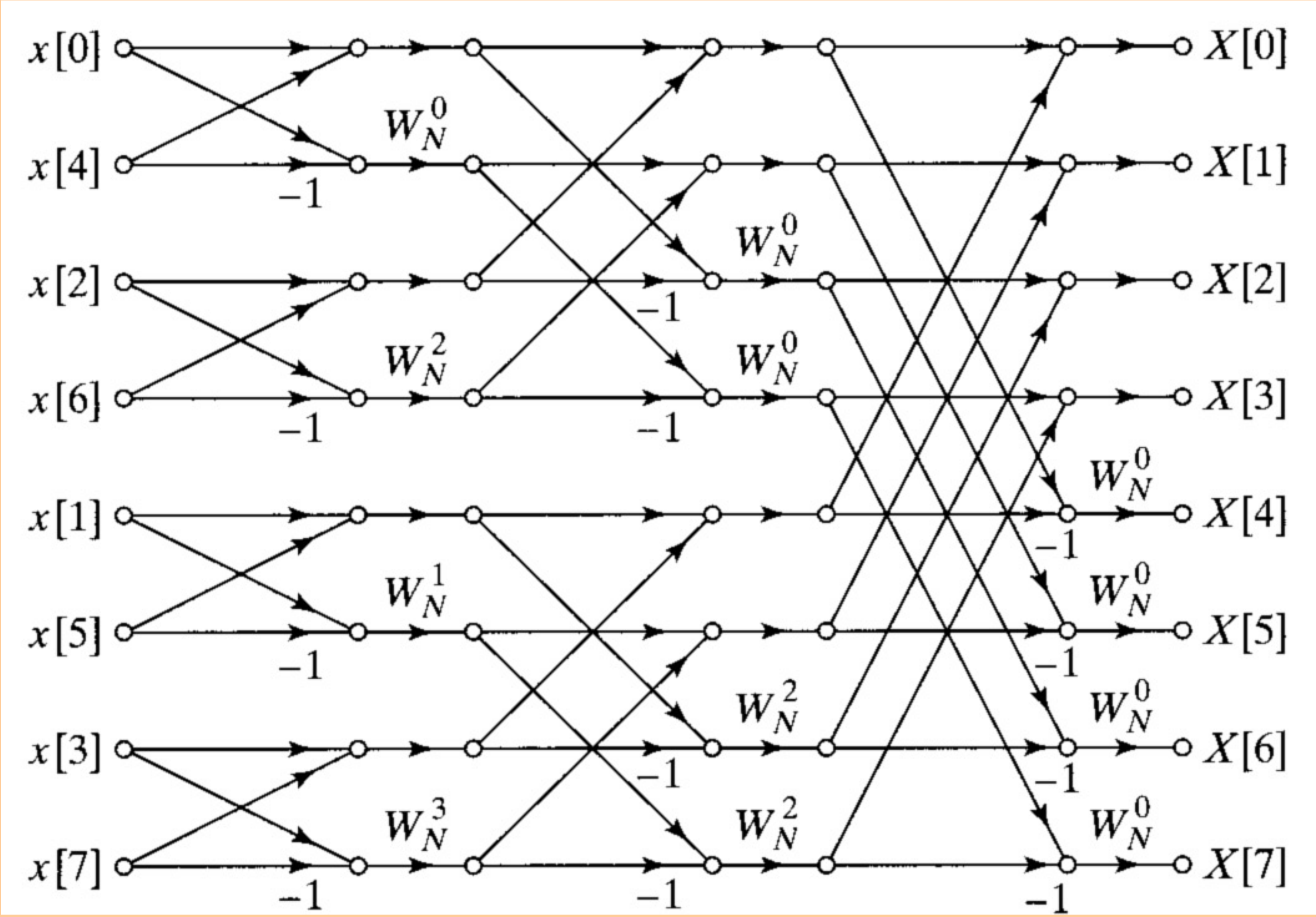
Alternate DIF FFT structures

- DIF structure with input natural, output bit-reversed (OSYP 9.22):



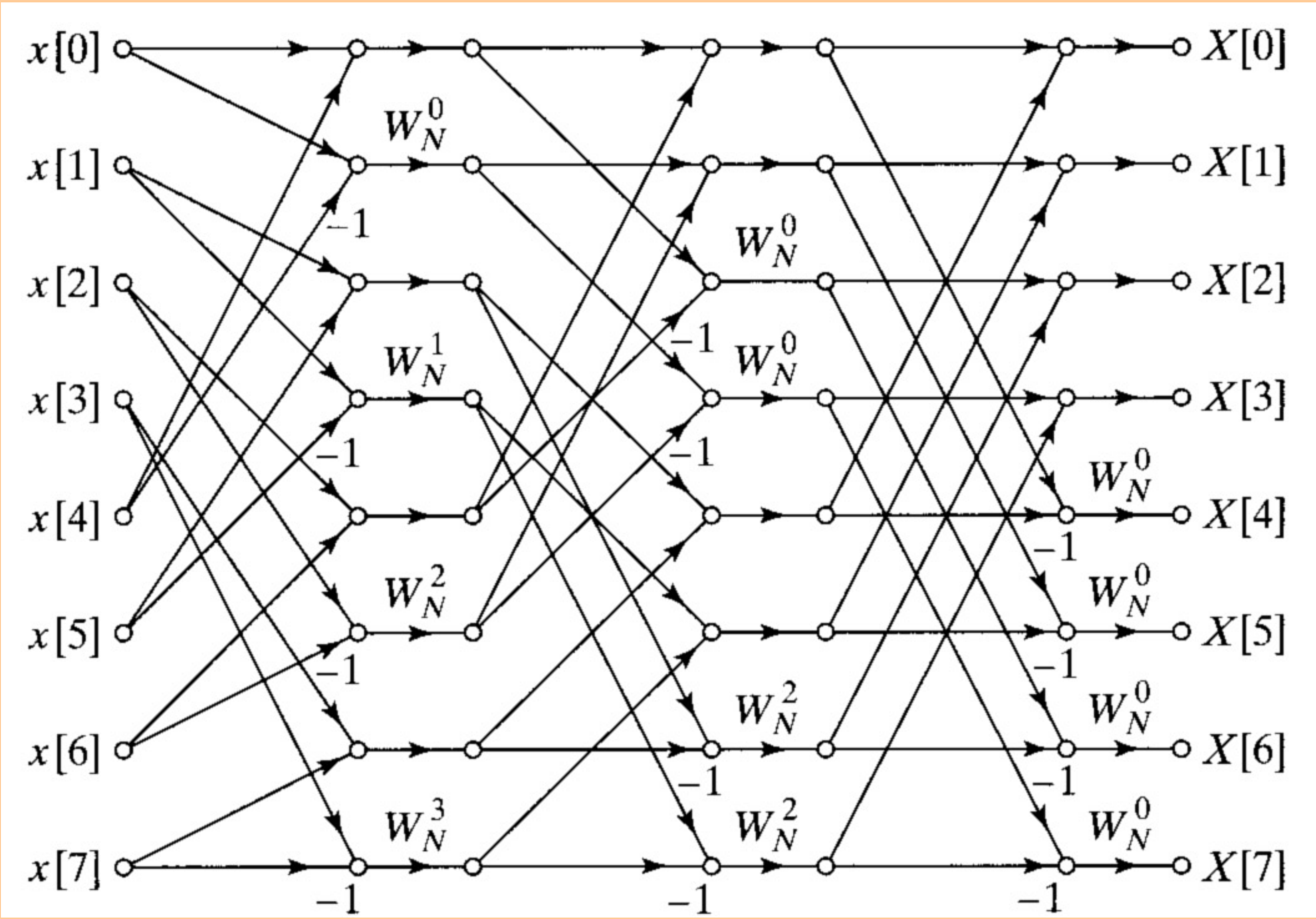
Alternate DIF FFT structures (continued)

- **DIF structure with input bit-reversed, output natural (OSB 9.22):**



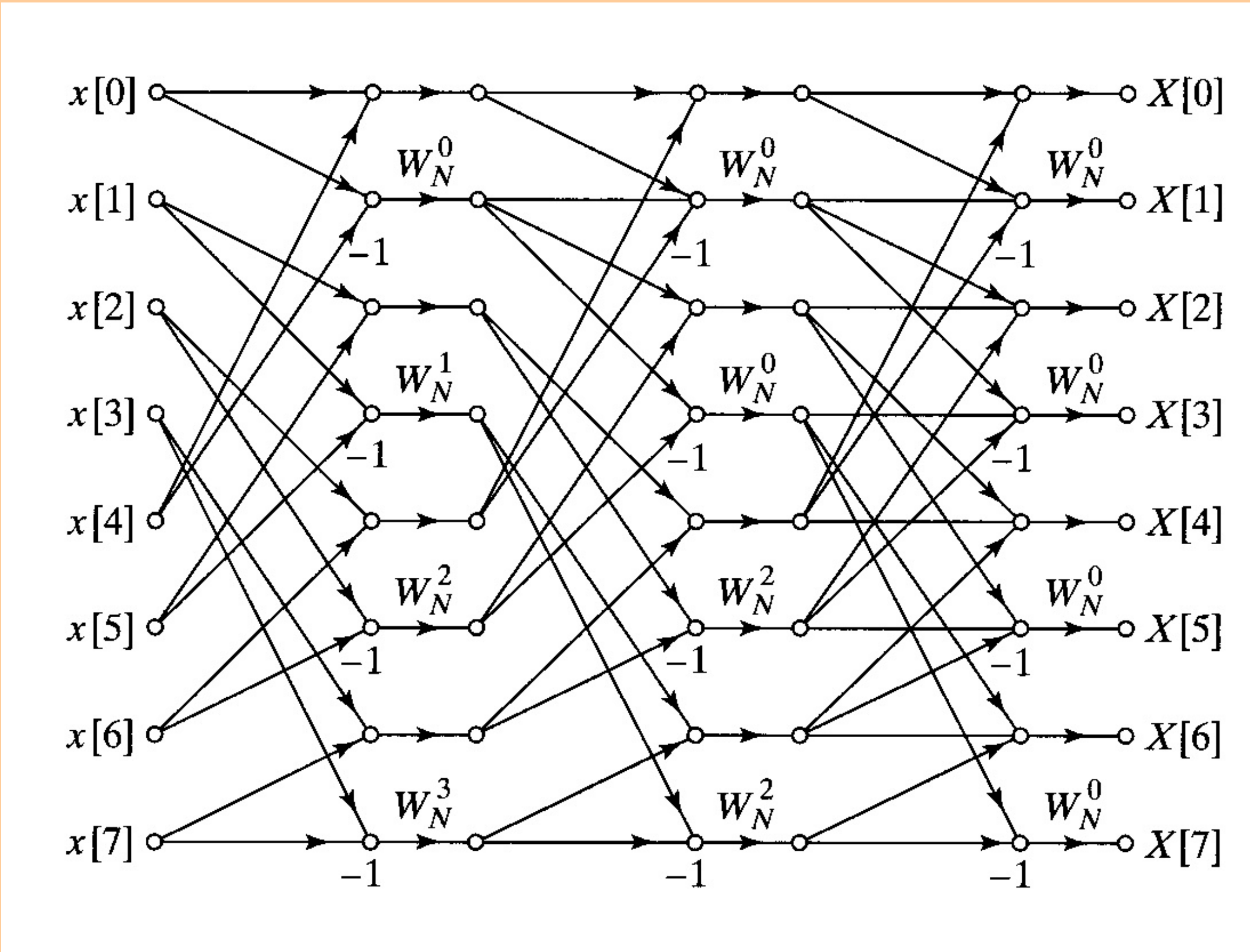
Alternate DIF FFT structures (continued)

- **DIF structure with both input and output natural (OSYP 9.24):**



Alternate DIF FFT structures (continued)

- DIF structure with same structure for each stage (OSYP 9.25):



FFT structures for other DFT sizes

- **Can we do anything when the DFT size *N* is not an integer power of 2 (the non-radix 2 case)?**
- **Yes! Consider a value of *N* that is not a power of 2, but that still is highly factorable …**

$$\text{Let } N = p_1 p_2 p_3 p_4 \ldots p_\nu; \;\; q_1 = N / p_1, \;\; q_2 = N / p_1 p_2, \text{etc.}$$

- **Then let**

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

$$= \sum_{r=0}^{q_1-1} x[p_1 r] W_N^{p_1 rk} + \sum_{r=0}^{q_1-1} x[p_1 r + 1] W_N^{(p_1 r + 1)k} + \sum_{r=0}^{q_1-1} x[p_1 r + 2] W_N^{(p_1 r + 2)k} + \ldots$$

Non-radix 2 FFTs (continued)

- **An arbitrary term of the sum on the previous panel is**

$$\sum_{r=0}^{q_1-1} x[p_1 r + l] W_N^{(p_1 r + l)k}$$

$$= \sum_{r=0}^{q_1-1} x[p_1 r + l] W_N^{p_1 r k} W_N^{lk} = W_N^{lk} \sum_{r=0}^{q_1-1} x[p_1 r + l] W_{q_1}^{rk}$$

- **This is, of course, a DFT of size q_1 of points spaced by p_1**

Non-radix 2 FFTs (continued)

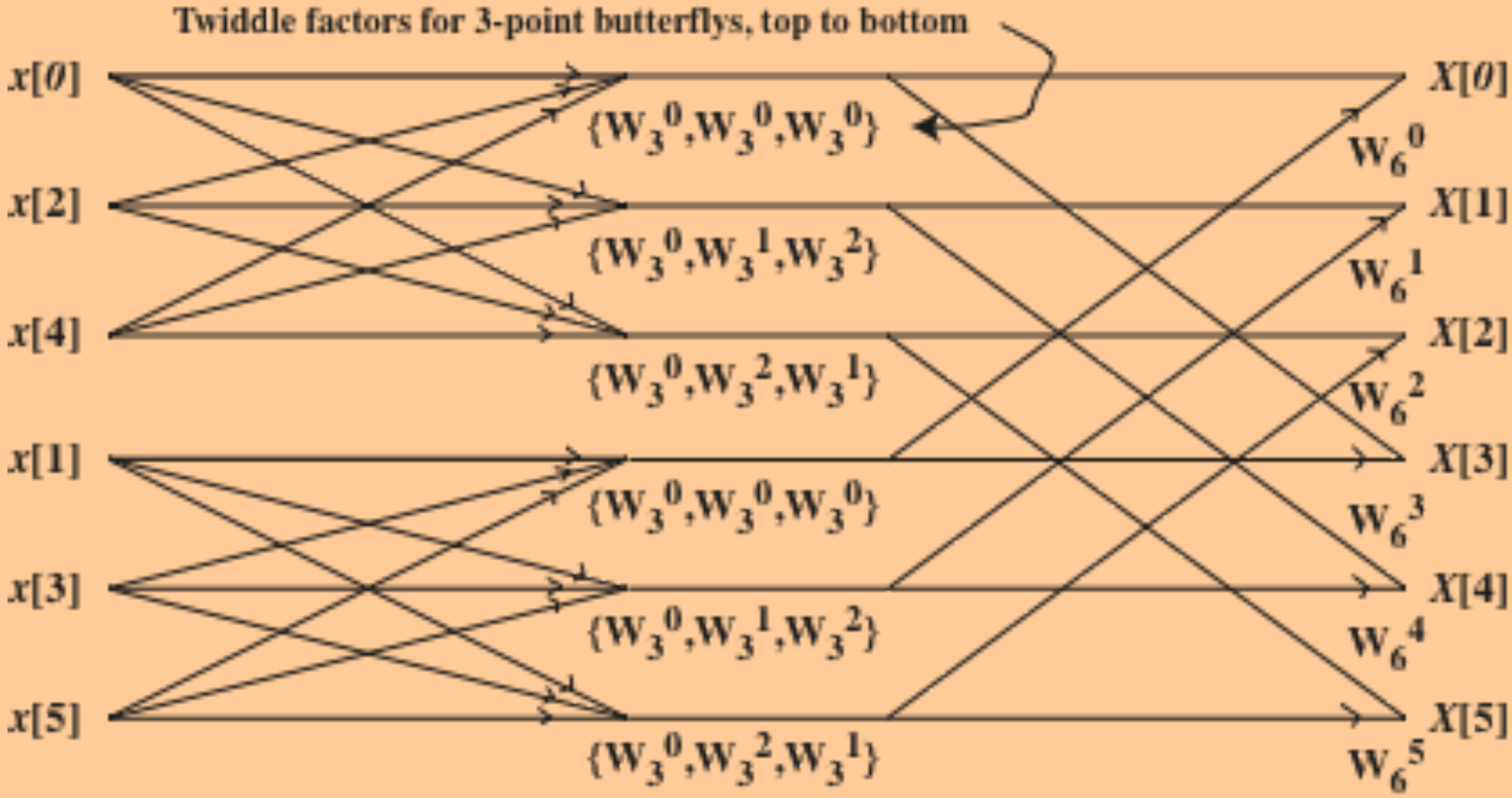
- **In general, for the first decomposition we use**

$$X[k] = \sum_{l=0}^{p_1-1} W_N^{lk} \sum_{r=0}^{q_1-1} x[p_1 r + l] W_{q_1}^{rk}$$

- **Comments:**
  - This procedure can be repeated for subsequent factors of *N*
  - The amount of computational savings depends on the extent to which *N* is "composite", able to be factored into small integers
  - Generally the smallest factors possible used, with the exception of some use of radix-4 and radix-8 FFTs

An example …. The 6-point DIT FFT

- $P_1 = 2; P_2 = 3;$ $$X[k] = \sum_{l=0}^{1} W_6^{lk} \sum_{r=0}^{2} x[2r+l] W_3^{rk}$$



Summary

- **This morning we considered a number of alternative ways of computing the FFT:**
  - Alternate implementation structures
  - The decimation-in-frequency structure
  - FFTs for sizes that are non-integer powers of 2
  - Using standard FFT structures for inverse FFTs

- **Starting on Monday we will begin to discuss digital filter implementation structures**