# Advanced Transform Methods

## Alternate FFT Structures

Andy Watson

# FFT Inverses and Variants

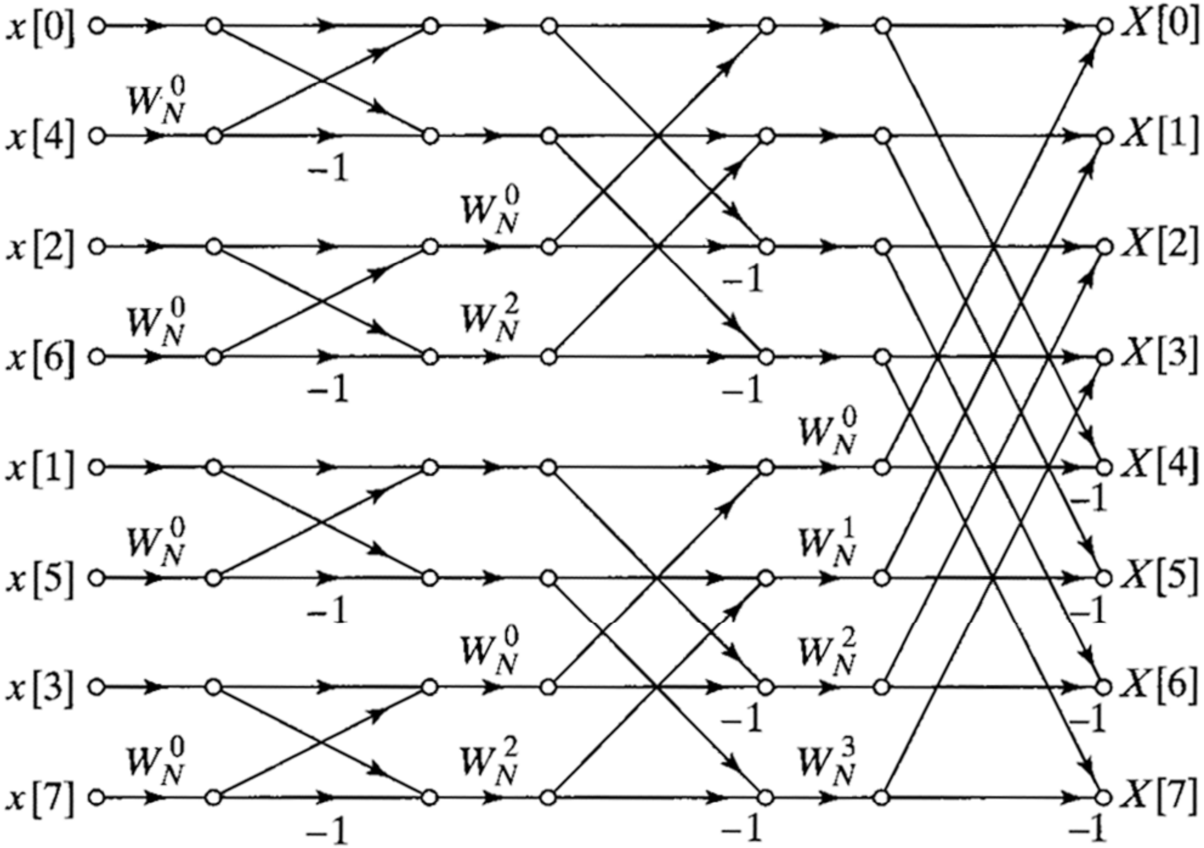
- Last lecture: basic decimation-in-time Cooley-Tuckey FFT alg., for DFT sizes that are powers of 2 (radix 2)
- Now: variations and extensions of the FFT algorithm:
  - Alternate forms of the FFT structure
  - Computation of the inverse DFT
  - The decimation-in-frequency FFT algorithm
  - FFT structures for DFT sizes that are not an integer power of 2

- Alternative FFT structures are possible simply by rearranging the branches of the signal flowgraph…

Acknowledgement: adapted from R M Stern (CMU)

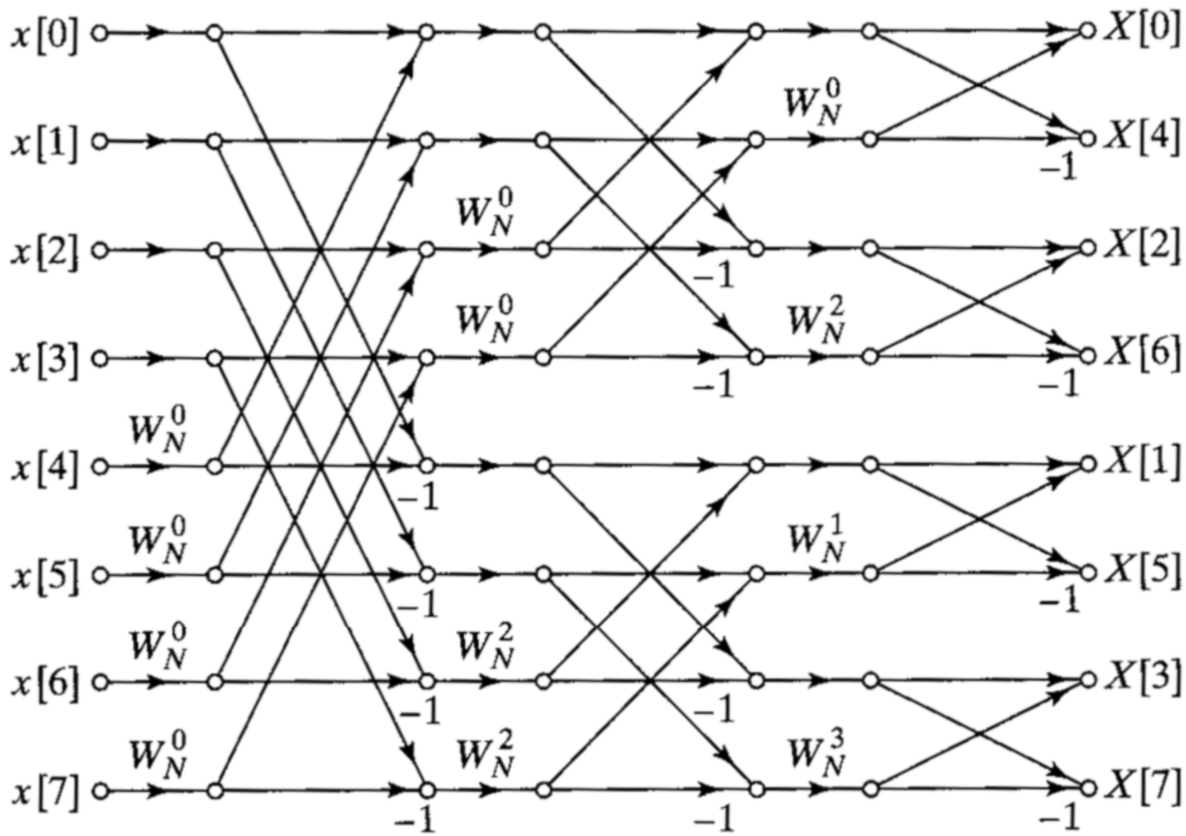
# Alternate DIT FFT structures (cont)

- DIT structure with input bit-reversed, output natural:



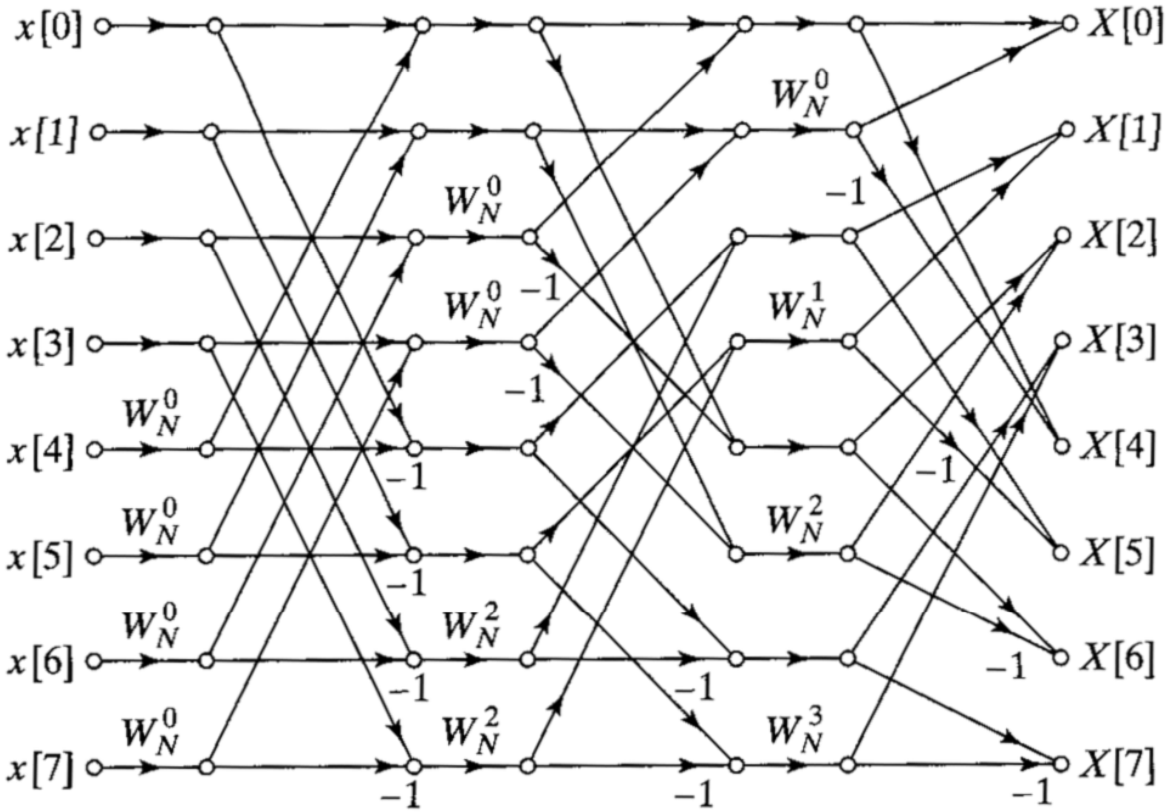
# Alternate DIT FFT structures (cont)

- DIT structure with input natural, output bit-reversed:



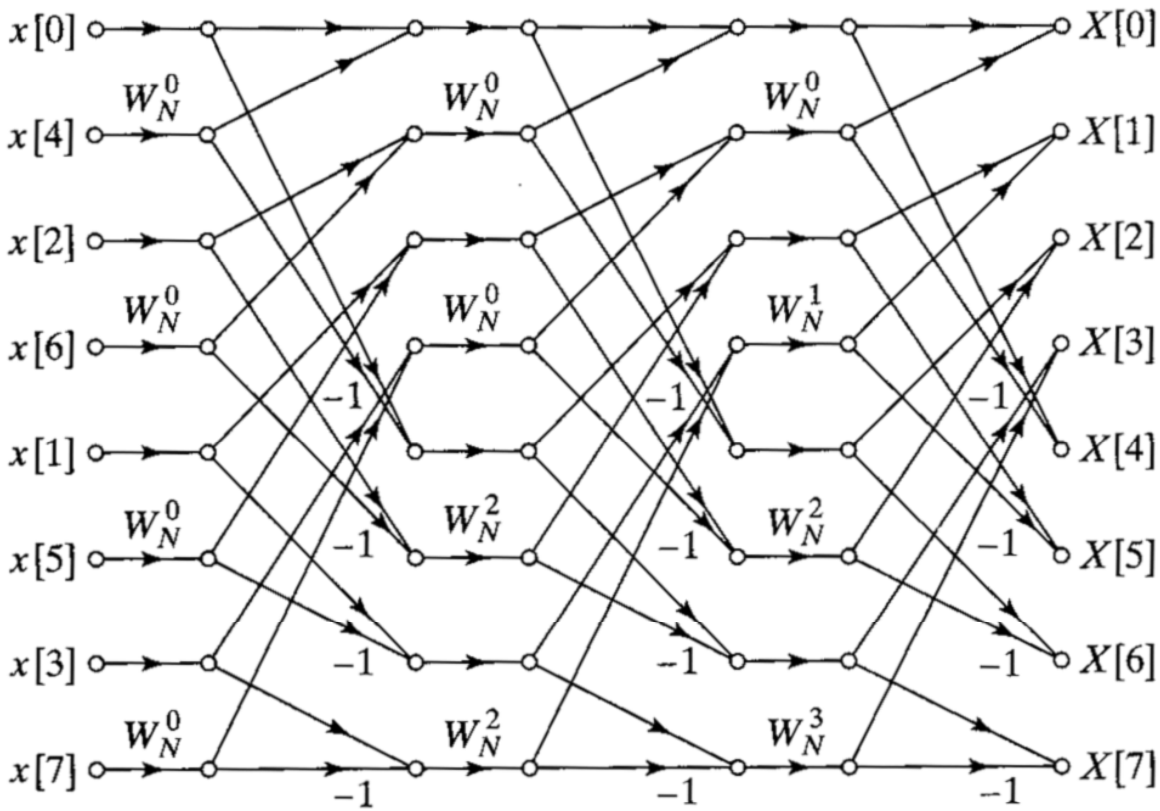
# Alternate DIT FFT structures (cont)

- DIT structure with both input and output natural:



# Alternate DIT FFT structures (cont)

- DIT structure with same structure for each stage:



# Comments on alternate FFT structures

- A method to avoid bit-reversal in filtering operations is:
  – Compute forward transform using natural input, bit-reversed output
  – 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
- Latter two topologies are now rarely used (previous two slides)

# Using FFTs for inverse DFTs

- We've always been talking about forward DFTs in our discussion about FFTs …. what about the inverse FFT?

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

- One way to modify FFT algorithm for the inverse DFT computation is:
  – Replace $W_N^k$ by $W_N^{-k}$ wherever it appears
  – Multiply final output by $1/N$
- This method has the disadvantage that it requires modifying the internal code in the FFT subroutine

Note sign in twiddle factor
in this lecture – common in FFT texts

# A better way to modify FFT code for inverse DFTs

- Taking the complex conjugate of both sides of the IDFT equation and multiplying by *N*:

$$Nx^*[n] = \sum_{k=0}^{N-1} X^*[k] W_N^{kn}; \text{or } x[n] = \tfrac{1}{N}\left[\sum_{k=0}^{N-1} X^*[k] W_N^{kn}\right]^*$$

- This suggests that we can modify the FFT algorithm for the inverse DFT computation by the following:
  - Complex conjugate the input DFT coefficients
  - Compute the *forward* FFT
  - Complex conjugate the output of the FFT and multiply by $1/N$
- This method has the advantage that the internal FFT code is undisturbed; it is widely 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 (cont)

- 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

# Decimation in frequency (cont)

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

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

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

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

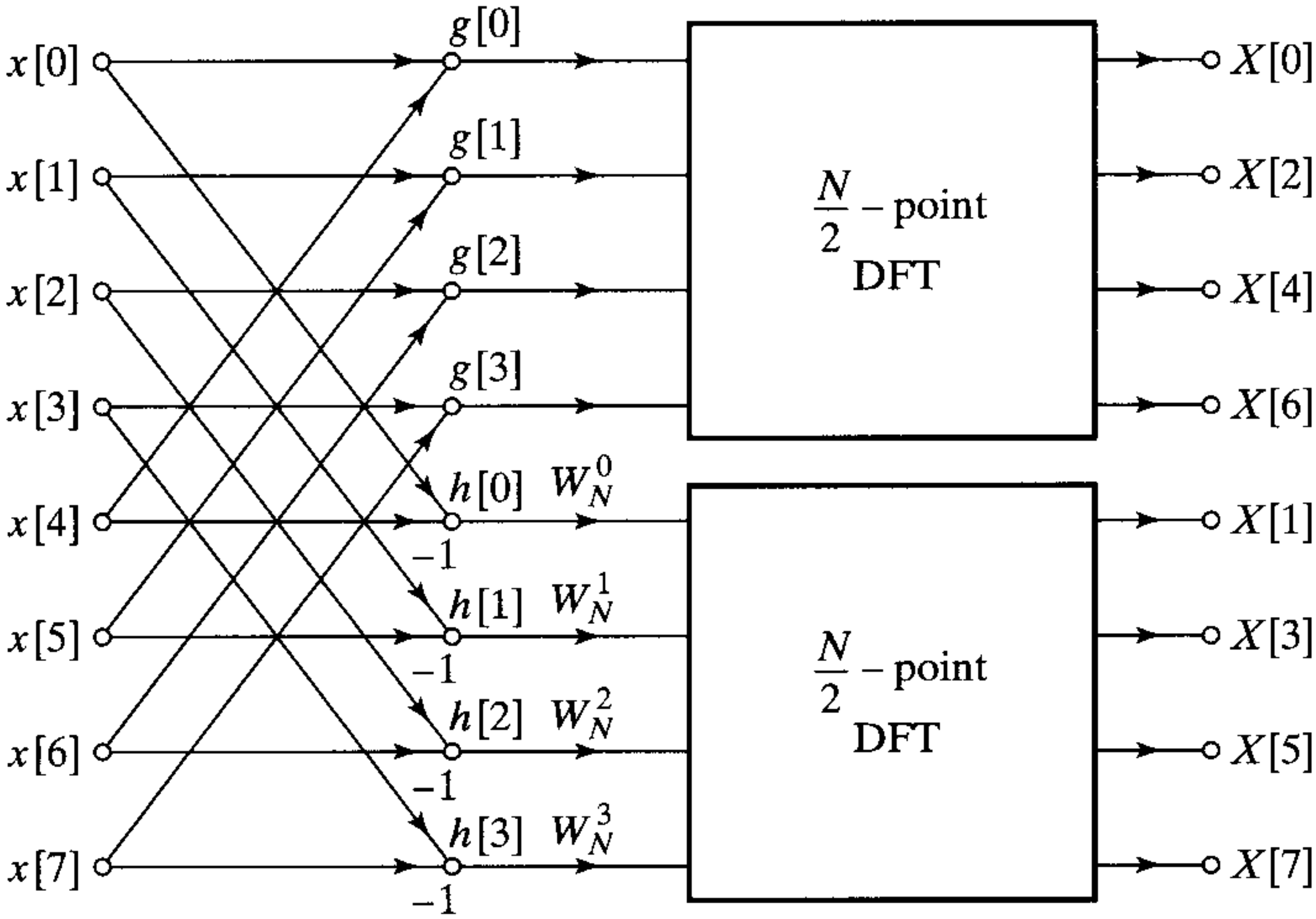
$$X[k] = \sum_{n=0}^{(N/2)-1} \left[ x[n] + (-1)^{2r} (-1) x[n+(N/2)] \right] W_N^{n(2r+1)}$$

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

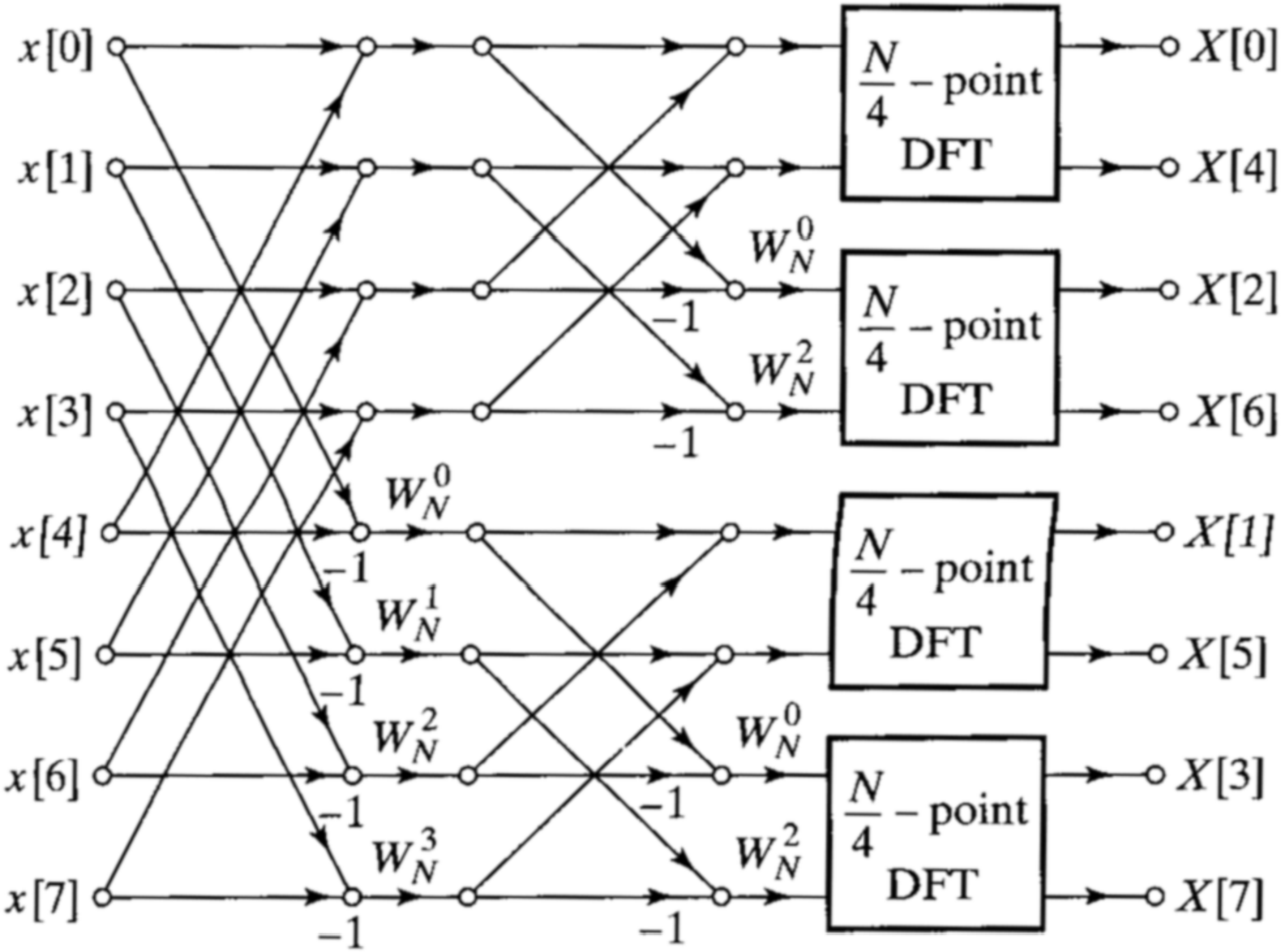
- These expressions are the *N/2*-point DFTs of

$x[n] + x[n+(N/2)]$ and $[x[n] - x[n+(N/2)]]W_N^n$

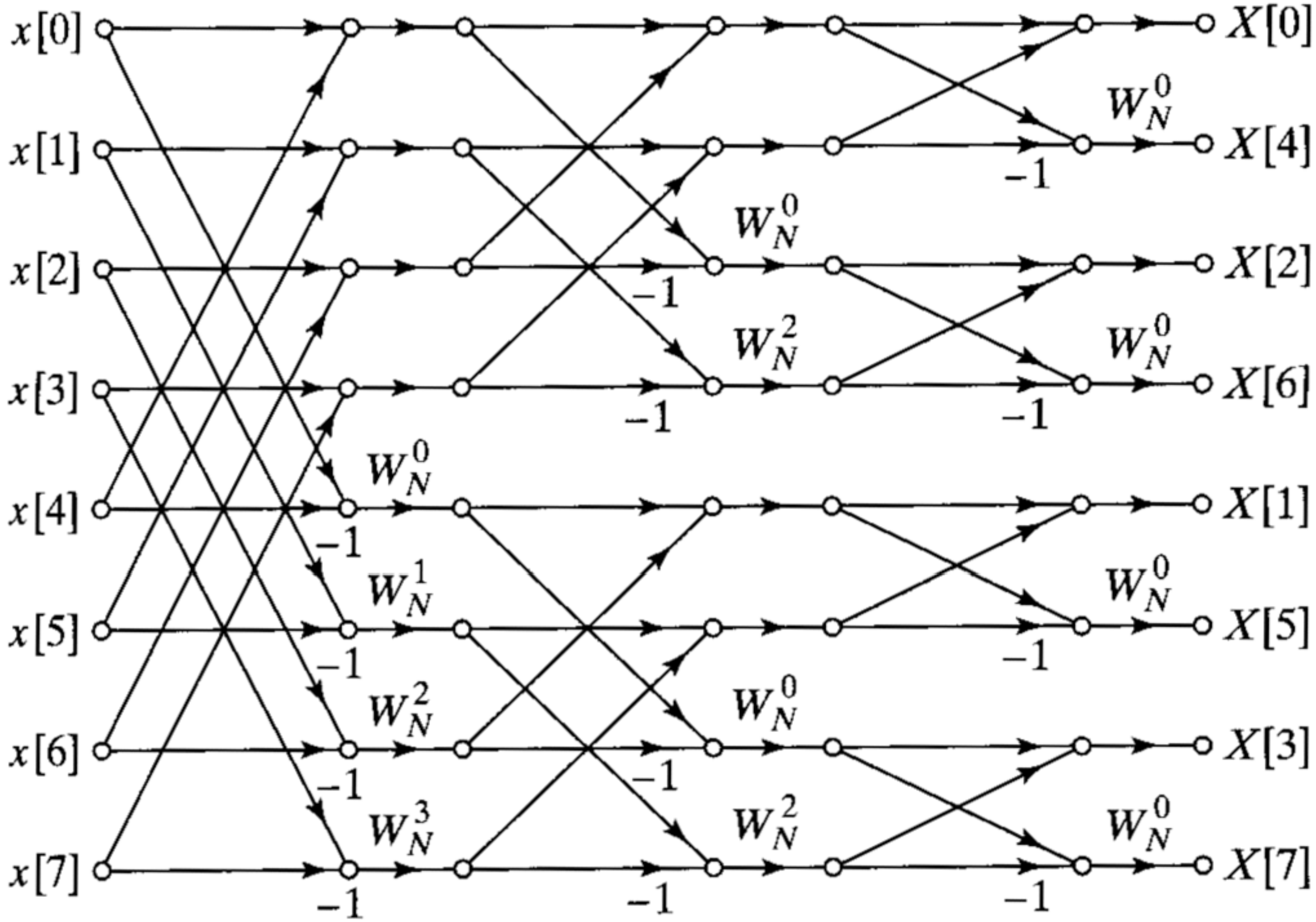
# These equations describe the following structure:



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

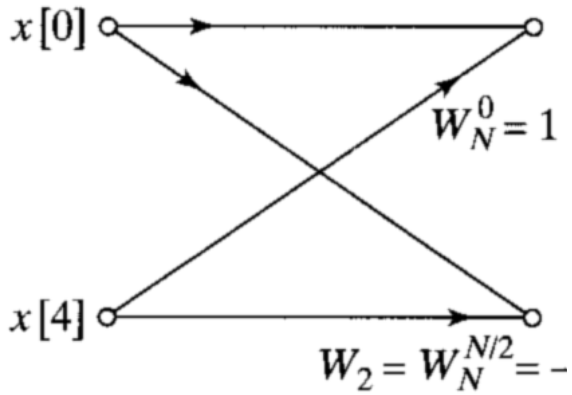


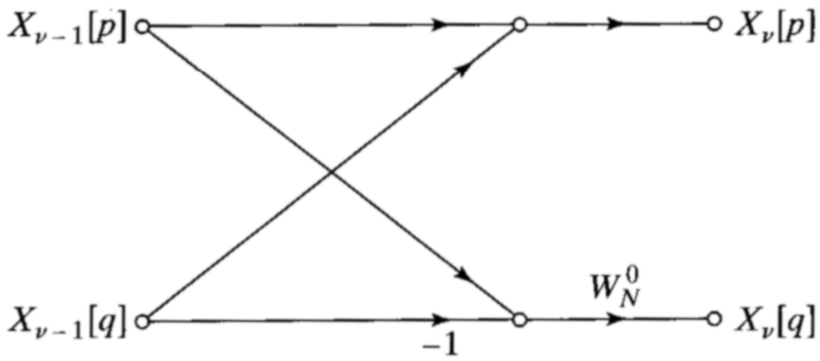
# … and replacing the *N/4*-point DFTs by butterflys we obtain



# 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:

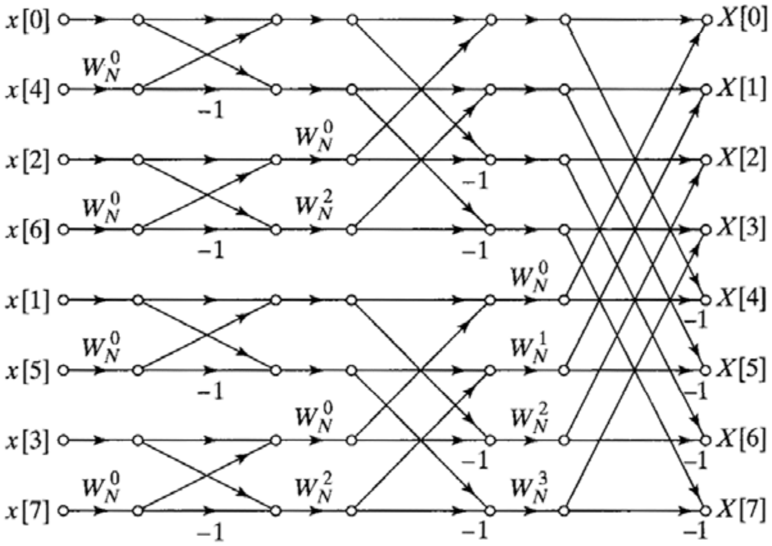


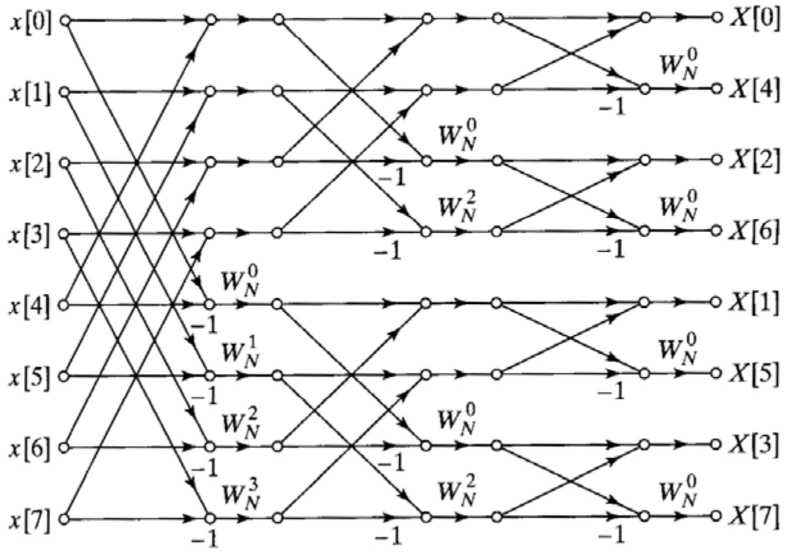
# DIF FFT is *transpose* of the DIT FFT

- Comparing DIT and DIF structures:

**DIT FFT structure:**



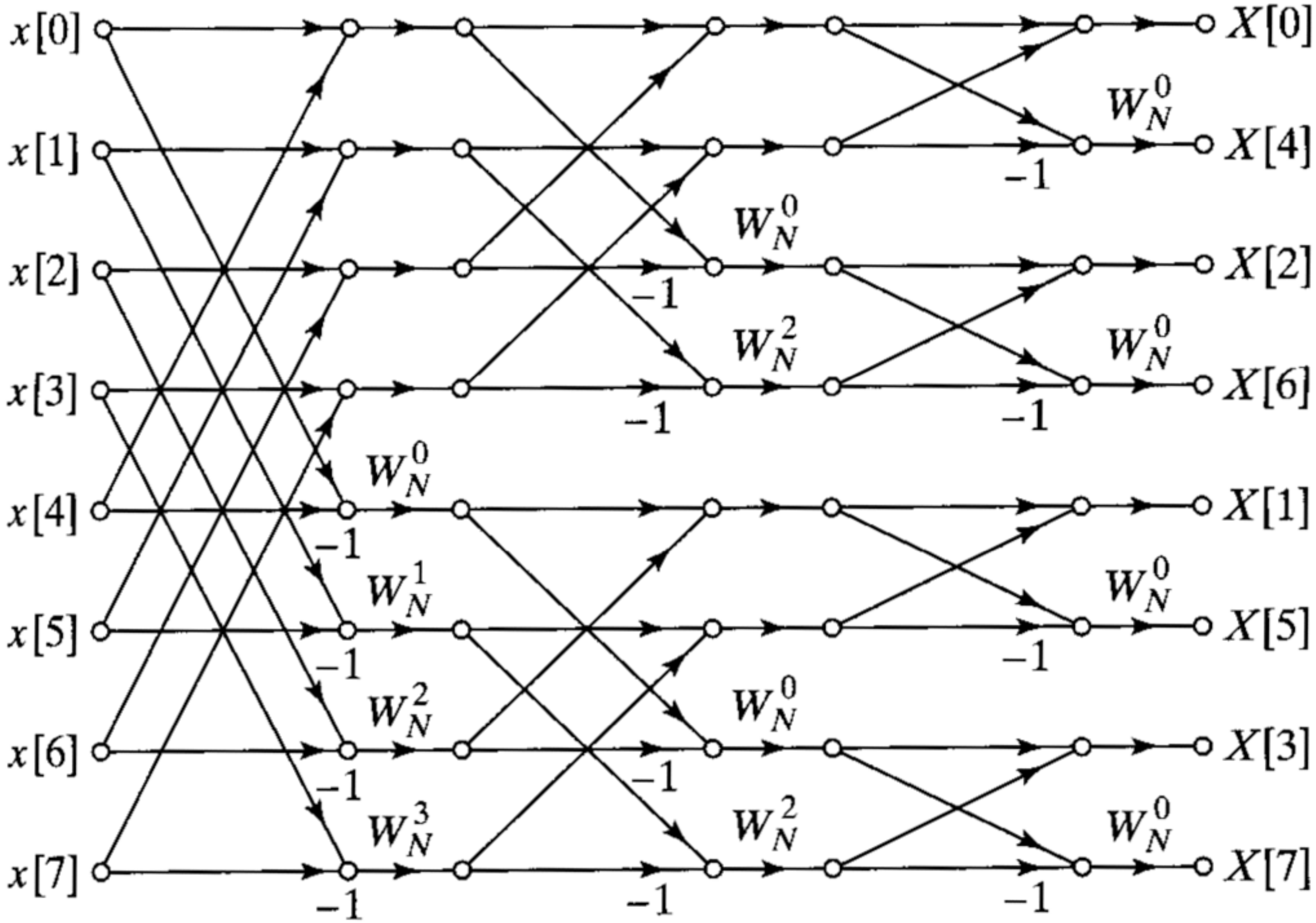

**DIF FFT structure:**




- Alternate forms for DIF FFTs are similar to DIT FFTs

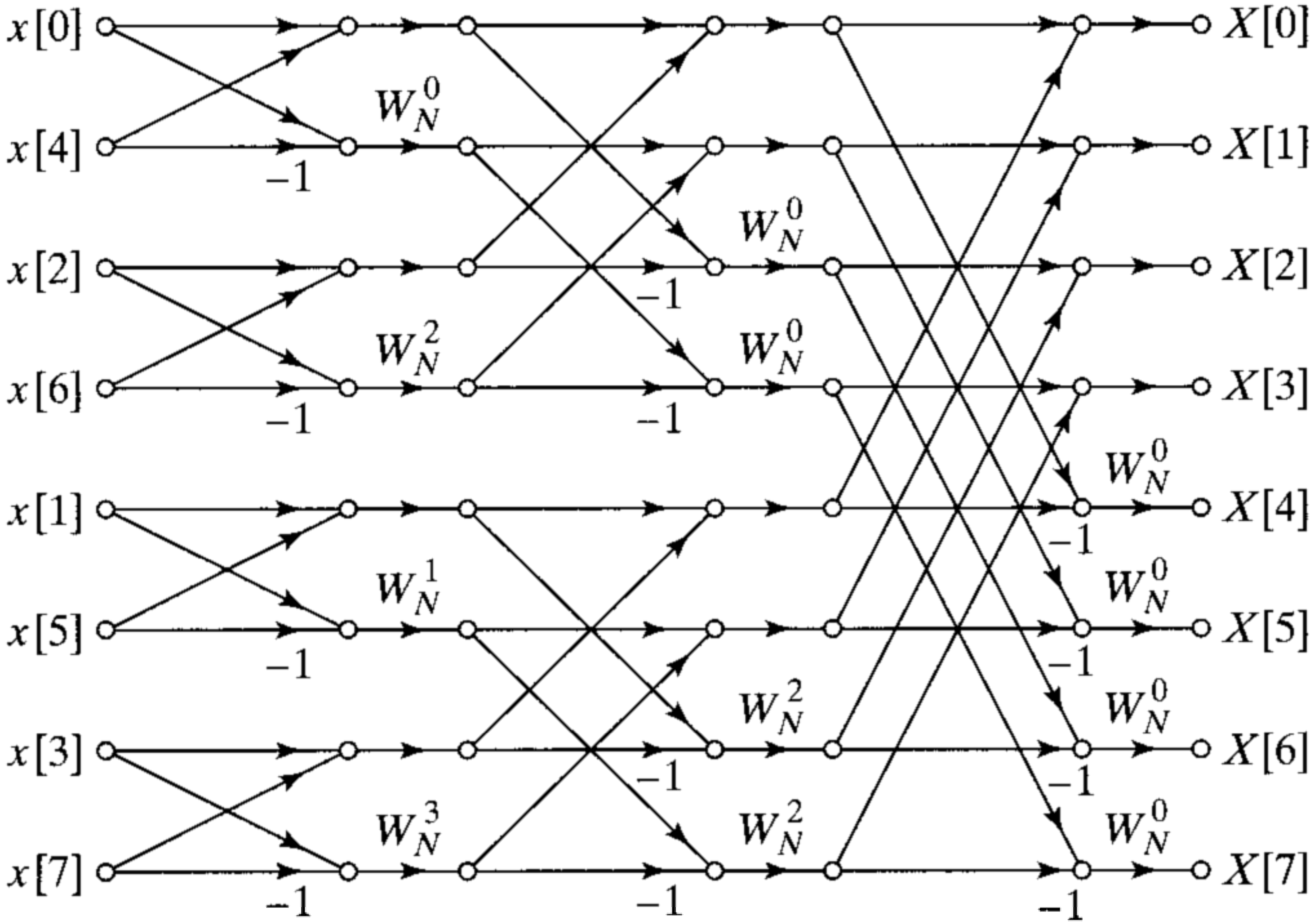
# Alternate DIF FFT structures

- DIF structure with input natural, output bit-reversed:



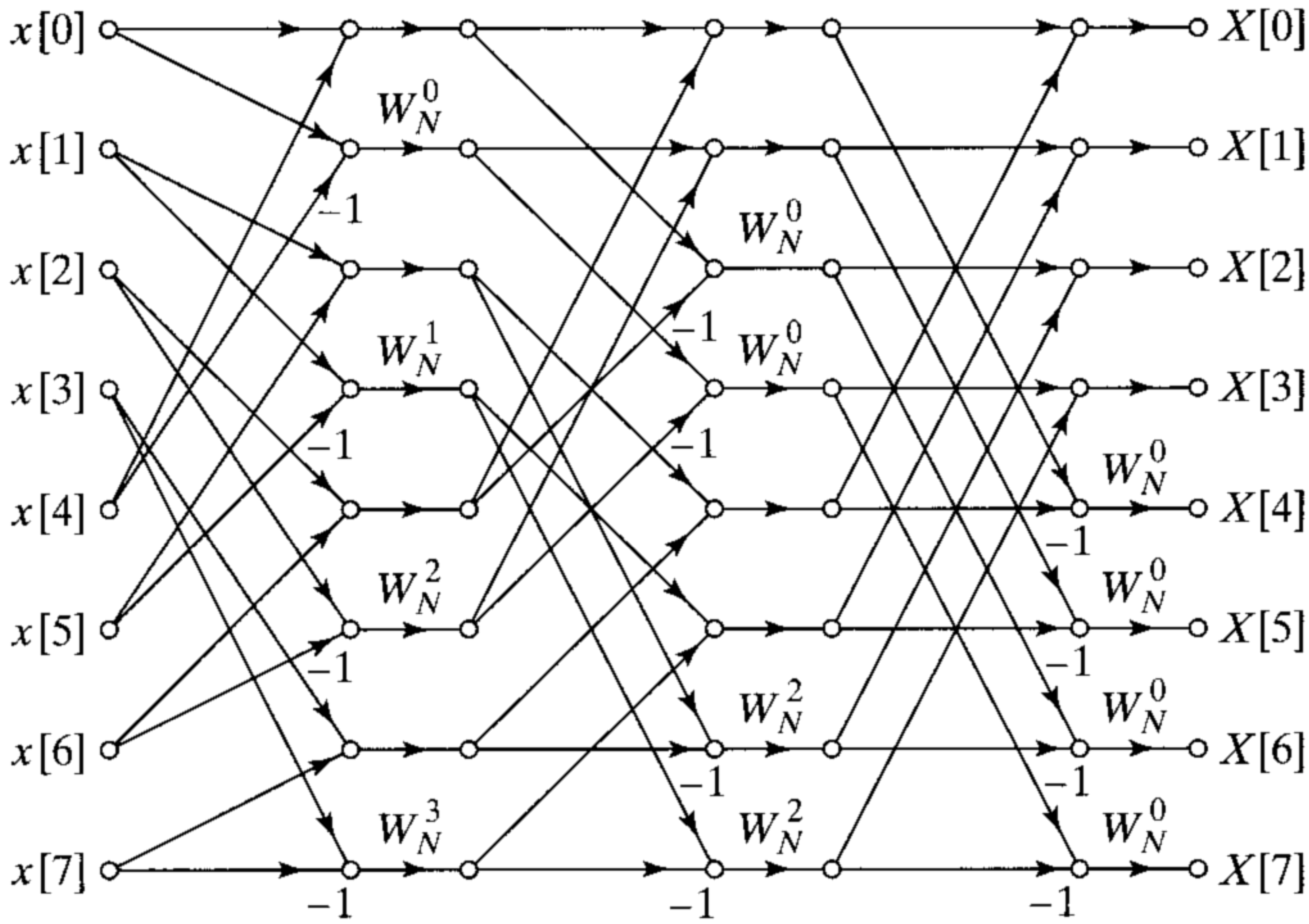
# Alternate DIF FFT structures (cont)

- DIF structure with input bit-reversed, output natural:



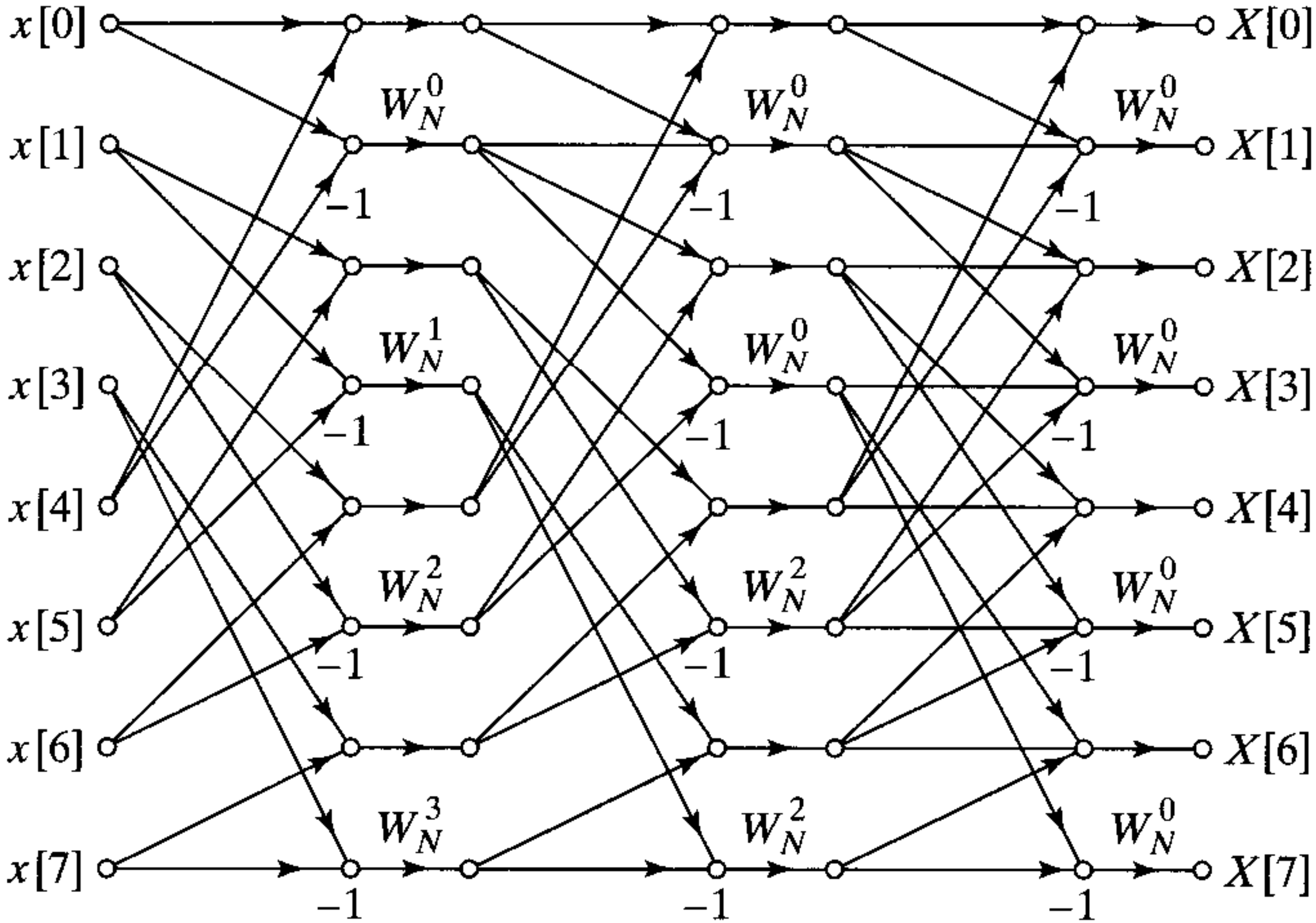
# Alternate DIF FFT structures (cont)

- DIF structure with both input and output natural:



# Alternate DIF FFT structures (cont)

- DIF structure with same structure for each stage:



# 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 …

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

- Then let

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

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

# Non-radix 2 FFTs (continued)

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

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

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

- This is 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

- $P_1 = 2; P_2 = 3;$

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