18-791 Lecture #18
FAST FOURIER TRANSFORM INVERSES AND 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
October 27, 2005

Introduction

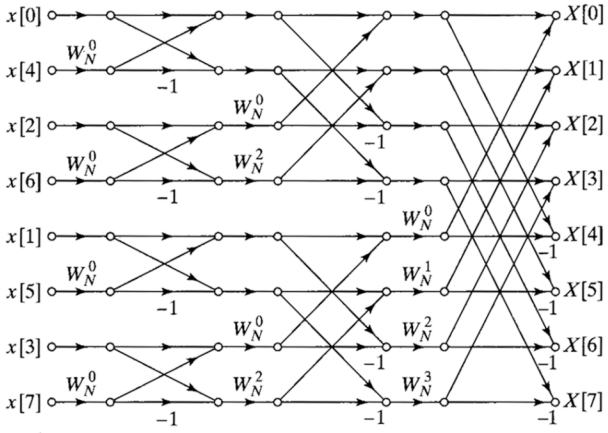
- **In our lecture last Tuesday we described and discussed the basic decimation-in-time Cooley-Tuckey fast Fourier transform algorithm for DFT sizes that are integer powers of 2 (radix 2)**
- **Today we will discuss some variations and extensions of the basic FFT algorithm:**
  - Alternate forms of the 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

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
- Consider the rearranged signal flow diagrams on the following panels …..

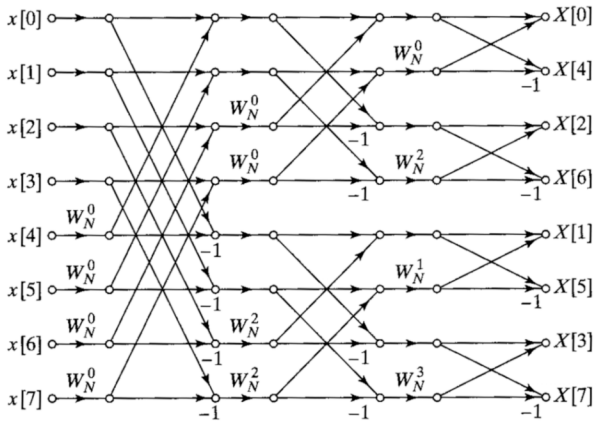
Alternate DIT FFT structures (continued)

- DIT structure with input bit-reversed, output natural (OSB 9.10)



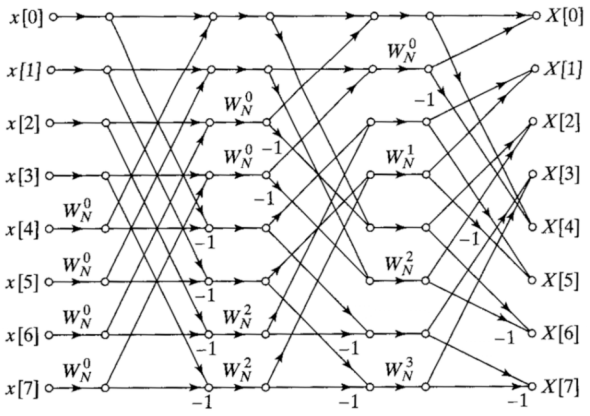
Alternate DIT FFT structures (continued)

- DIT structure with input natural, output bit-reversed (OSB 9.14):



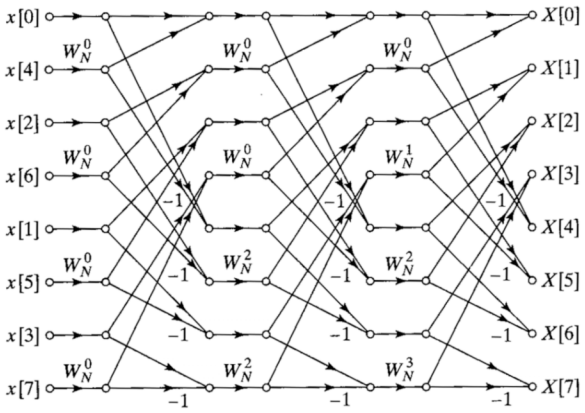
Alternate DIT FFT structures (continued)

- DIT structure with both input and output natural (OSB 9.15):



Alternate DIT FFT structures (continued)

- **DIT structure with same structure for each stage (OSB 9.16):**

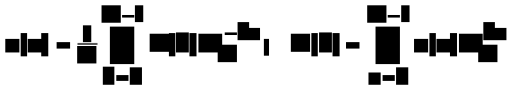


Comments on alternate FFT structures

- **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 OSB 9.15 and 9.16) are now rarely used**

Using FFTs for inverse DFTs

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



- **One way to modify FFT algorithm for the inverse DFT computation is:**
  - Replace  by  wherever it appears
  - Multiply final output by
- **This method has the disadvantage that it requires modifying the internal code in the FFT subroutine**

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*:**

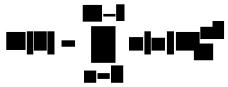
- **This suggests that we can modify the FFT algorithm for the inverse DFT computation by the following:**
  - Complex conjugate the input DFT coefficients
  - Compute the *forward* FFT
  - Complex conjugate the output of the FFT and multiply by 
- **This method has the advantage that the internal FFT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

The decimation-in-frequency (DIF) FFT algorithm

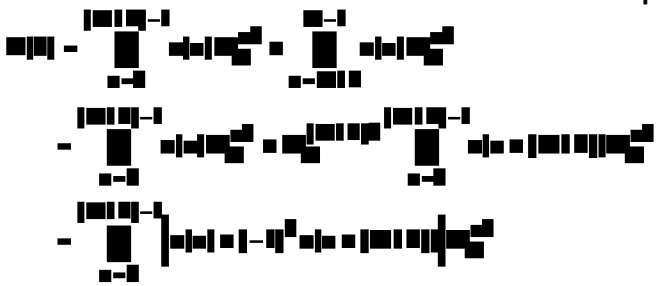
- Introduction: Decimation in frequency is an alternate way of developing the FFT algorithm
- It is different from decimation in time in its development, although it leads to a very similar structure

The decimation in frequency FFT (continued)

- Consider the original DFT equation ….



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

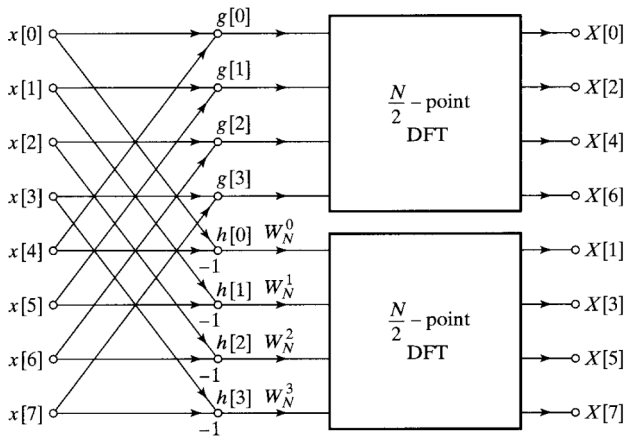


- Note that these are not *N/2*-point DFTs

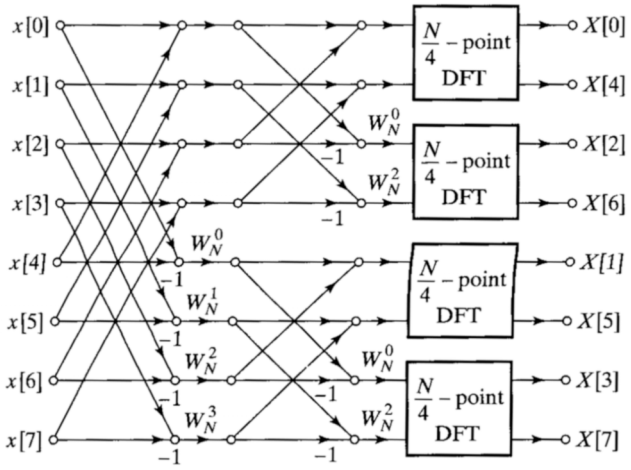
Continuing with decimation in frequency ...

- For *k* even, let
- For *k* odd, let
- These expressions are the *N*/2-point DFTs of

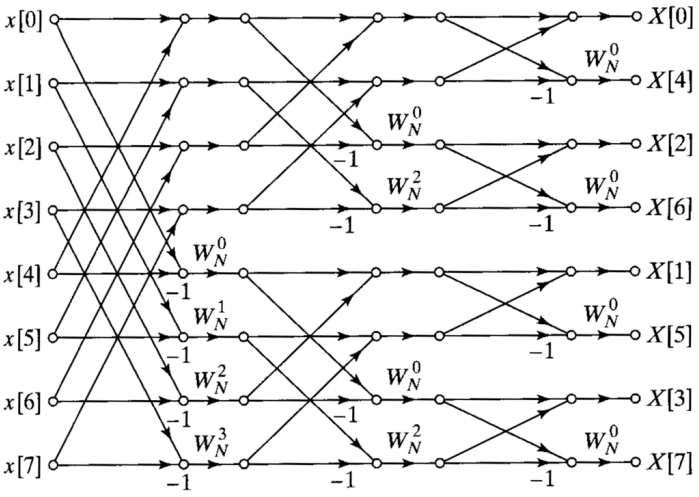
These equations describe the following structure:



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

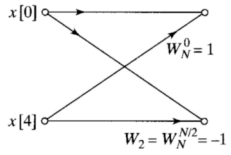


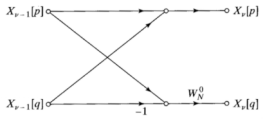
… and replacing the *N/4*-point DFTs by butterflys we obtain



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:**



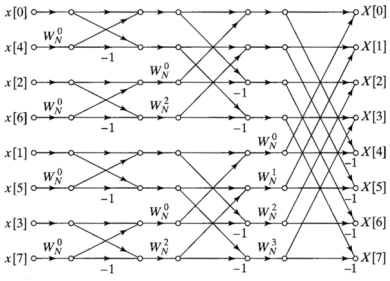


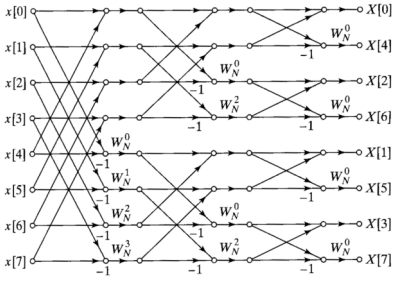
- **Comment:**
  - We will revisit transposed forms again in our discussion of filter implementation

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

- **Comparing DIT and DIF structures:**

DIT FFT structure: **DIF FFT structure:**

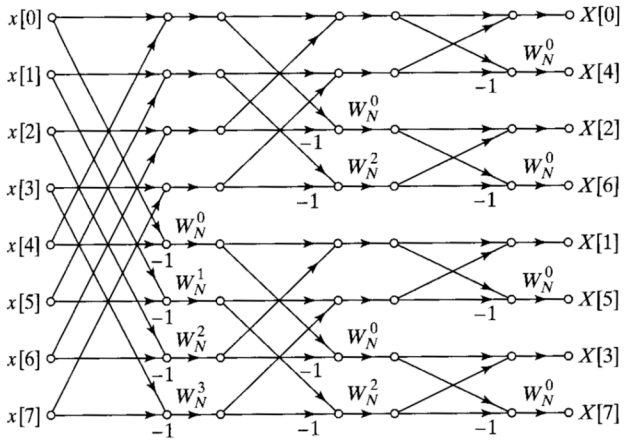




- **Alternate forms for DIF FFTs are similar to those of DIT FFTs**

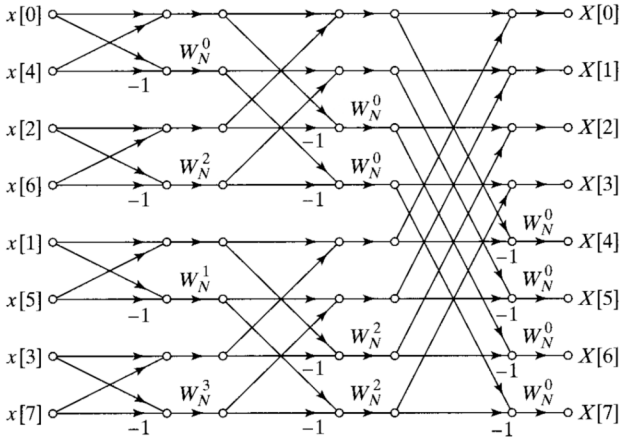
Alternate DIF FFT structures

- DIF structure with input natural, output bit-reversed (OSB 9.20)



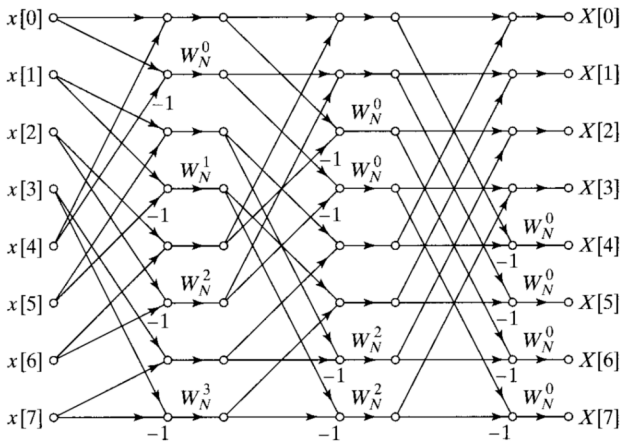
Alternate DIF FFT structures (continued)

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



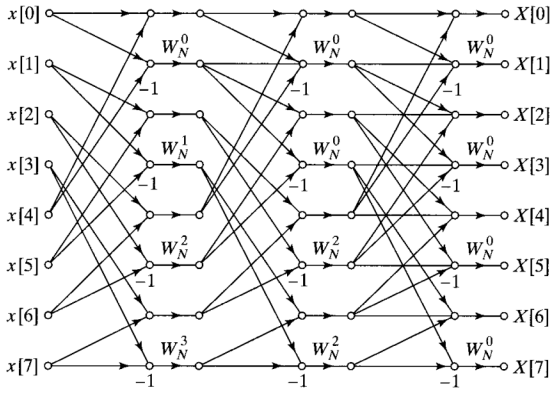
Alternate DIF FFT structures (continued)

- DIF structure with both input and output natural (OSB 9.23):



Alternate DIF FFT structures (continued)

- DIF structure with same structure for each stage (OSB 9.24):



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 …
- Then let

Non-radix 2 FFTs (continued)

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

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

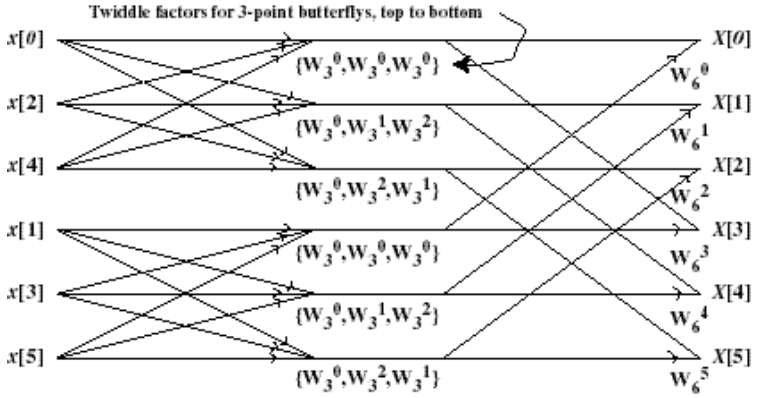
Non-radix 2 FFTs (continued)

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

- **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

- $P_1 = 2; P_2 = 3;$ 



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 Tuesday we will begin to discuss digital filter implementation structures**