Fast Fourier Transform

Dr Yvan Petillot

FFT Algorithms

- Developed by Cooley and Tukey in 1965
- Revolutionised signal processing and paved the way for DSP.
- Various forms exist:
  - Decompistion in Frequency (DIF)
  - Decomposition in Time (DIT)
  - Radix-4, Radix-2
- Critical for filtering and convolution

Idea behind FFT

DFT:

$$X(k)=\sum_{n=0}^{N-1} x_p(n)\ e^{-j2\pi\frac{kn}{N}} = \sum_{n=0}^{N-1} x_p(n)\ W_N^{nk}, \qquad 0 \le k \le N-1$$

Requires N^2 multiplication and N^2 addition.

Can this be reduced by finding more efficient ways of calculating X(k)?

Subdividing the DFT into smaller DFTs is the solution!

Decimation in time (DIT)

DFT:

$$X(k)=\sum_{n=0}^{N-1} x(n)\ e^{-j2\pi\frac{kn}{N}} = \sum_{n=0}^{N-1} x(n)\ W_N^{nk}, \qquad 0 \le k \le N-1$$

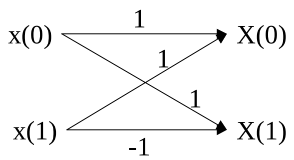
2 Points DFT:

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

$$W_2 = e^{-\frac{j2\pi}{2}} = -1$$

$$X(0) = x(0) + x(1)$$

$$X(1) = x(0) - x(1)$$



No complex multiply!

Butterfly structure

Decimation in time 4 points DFT

4 Points DFT:

$$X(k) = \sum_{n=0}^{4} x(n)\ W_4^{nk} = x(0)W_4^{0} + x(1)W_4^{1k} + x(2)W_4^{2k} + x(3)W_4^{3k}$$

$$W_4^{0k} = W_2^{0k}$$

$$W_4^{2k} = W_2^{1k}$$

$$X(k) = x(0)W_2^{0} + x(2)W_2^{1k} + W_4^{k}\left[x(1)W_4^{0k} + x(3)W_4^{2k}\right]$$

$$X(k) = DFT_2\left[x(n)\right]_{n\,even} + W_4^{k}\ DFT_2\left[x(n)\right]_{n\,odd}$$

$$X(k) = X_1(k) + W_4^{k}X_2(k)$$

where $x_1(n) = x(2n)$ and $x_2(n) = x(2n+1)$

Decimation in time 4 points DFT

4 Points DFT:

$$X(k) = X_1(k) + W_4^{k}X_2(k)$$

$$X_1(k+2) = X_1(k)$$

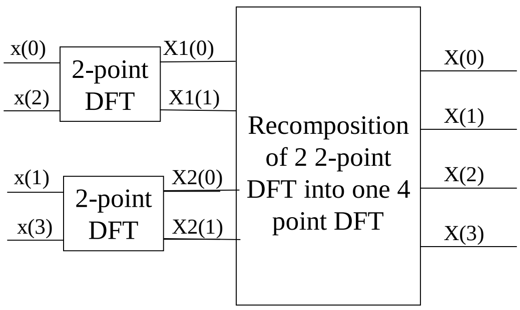
$$X_2(k+2) = X_2(k)$$

$$X(0) = X_1(0) + X_2(0)$$

$$X(1) = X_1(1) + W_4^{1}X_2(1)$$

$$X(2) = X_1(0) - X_2(0)$$

$$X(3) = X_1(1) - W_4^{1}X_2(1)$$



Decimation in time: Example

Example: x(n) = [0 1 2 3]

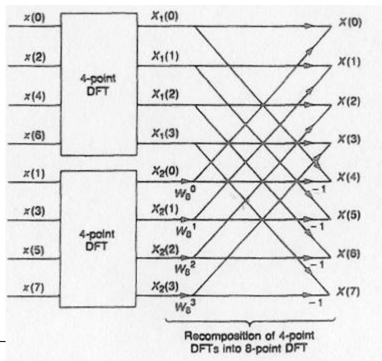
$$X_1(0) = x(0) + x(2) = 2$$
$$X_1(1) = x(0) - x(2) = -2$$
$$X_2(0) = x(1) + x(3) = 4$$
$$X_2(1) = x(1) - x(3) = -2$$

$$W_4^1 = e^{-\frac{j2\pi}{4}} = e^{-\frac{j\pi}{2}} = -j$$

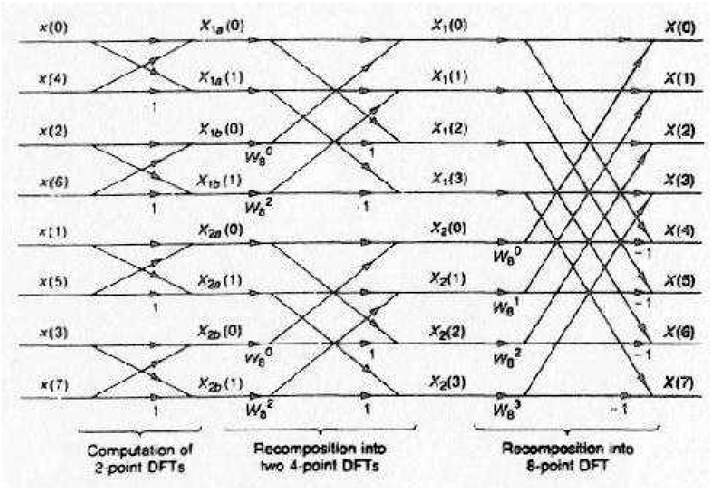
$$X(0) = X_1(0) + X_2(0) = 6$$
$$X(1) = X_1(1) + W_4^1 X_2(1) = -2 + 2j$$
$$X(2) = X_1(0) - X_2(0) = -2$$
$$X(3) = X(1) = X_1(1) - W_4^1 X_2(1) = -2 - 2j$$

Decimation in time:8 points case

$$X(k) = \sum_{n=0}^{7} x(n) W_8^{nk} \quad , \quad k = 0,1,....,7$$
$$X(k) = x(0)W_8^{0k} + x(1)W_8^{1k} + x(2)W_8^{2k} + x(3)W_8^{3k} + x(4)W_8^{4k} + x(5)W_8^{5k} + x(6)W_8^{6k} + x(7)W_8^{7k}$$
$$X(k) = x(0)W_8^{0k} + x(2)W_8^{2k} + x(4)W_8^{4k} + x(6)W_8^{6k} + W_8^{1k}\left[x(1)W_8^{0k} + x(3)W_8^{2k} + x(5)W_8^{4k} + x(7)W_8^{6k}\right]$$
$$W_8^{2nk} = W_4^{nk}$$
$$X(k) = x(0)W_4^{0k} + x(2)W_4^{k} + x(4)W_4^{2k} + x(6)W_4^{3k} + W_8^{1k}\left[x(1)W_4^{0k} + x(3)W_4^{1k} + x(5)W_4^{2k} + x(7)W_4^{3k}\right]$$
$$X(k) = X_1(k) + W_8^k X_2(k)$$
$$W_8^{k+4} = -W_8^k$$

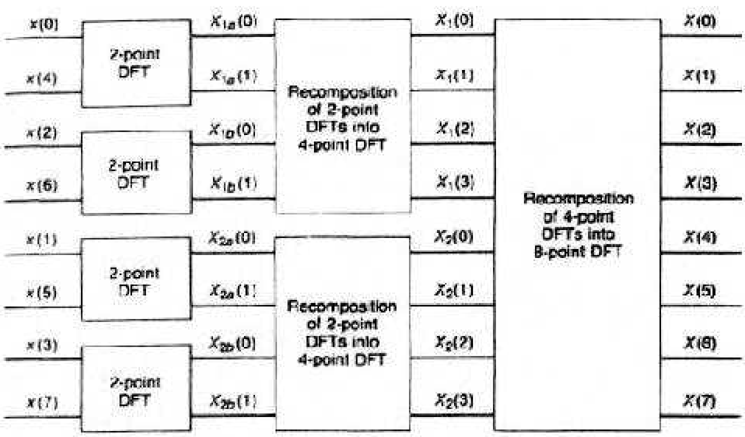


Decimation in time:8 points case



(b) Flow graph of 8-point DFT computed by recomposition of 2-point and 4-point DFTs

Decimation in time:8 points case



(a) Block diagram of 8-point DFT computed by recomposition of 2-point and 4-point DFTs

Decimation in time: general case

$$X(k)=\sum_{n=0}^{N-1}x(n)\ e^{-j2\pi\frac{kn}{N}}=\sum_{n=0}^{N-1}x(n)\ W_N^{nk}$$
$$=\sum_{n\,\text{even}}x(n)\ W_N^{nk}+\sum_{n\,\text{odd}}x(n)\ W_N^{nk}$$
$$=\sum_{l=0}^{N/2-1}x(2l)\ W_N^{2lk}+\sum_{l=0}^{N/2-1}x(2l+1)\ W_N^{(2l+1)k}$$

BUT :

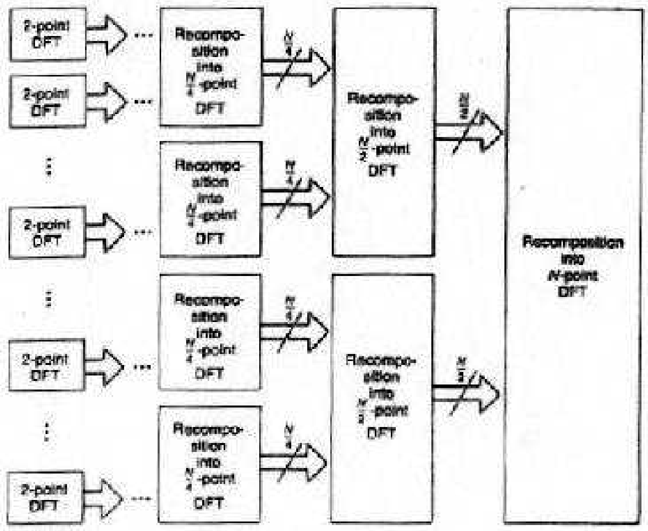
$$W_N^{2k}=e^{-\frac{j2\pi}{N}2lk}=e^{-\frac{j2\pi}{N/2}lk}=W_{N/2}^{lk}$$

$$X(k)=\sum_{l=0}^{N/2-1}x(2l)\,W_{N/2}^{lk}+W_N^{k}\sum_{l=0}^{N/2-1}x(2l+1)\ W_{N/2}^{lk}$$
$$=\sum_{l=0}^{N/2-1}x_1(l)\,W_{N/2}^{lk}+W_N^{k}\sum_{l=0}^{N/2-1}x_2(l)\ W_{N/2}^{lk}$$
$$=X_1(k)+W_N^k\ X_2(k)$$

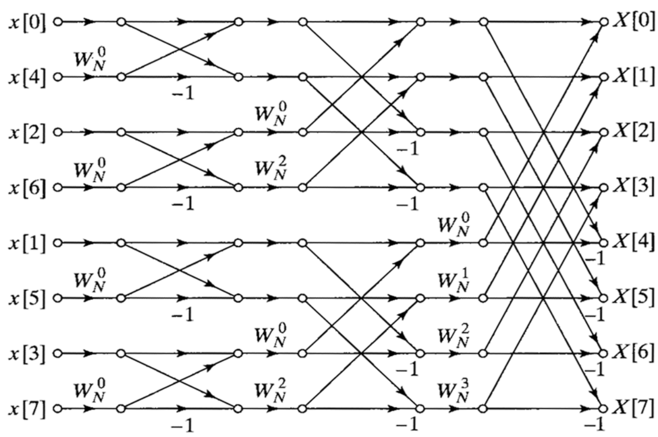
$$X_1\left[k+\frac{N}{2}\right]=X_1(k)\quad,\quad X_2\left[k+\frac{N}{2}\right]=X_2(k)$$
$$W_N^{k+N/2}=W_N^kW_N^{N/2}=-W_N^k$$

$$x_1(n)=x(2n),\quad n=0,1,\dots,N/2-1$$
$$x_2(n)=x(2n+1),\quad n=0,1,\dots,N/2-1$$

Decimation in time: general case



Decimation in time: general case



Using FFTs for inverse DFTs

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

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

- 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

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*:

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

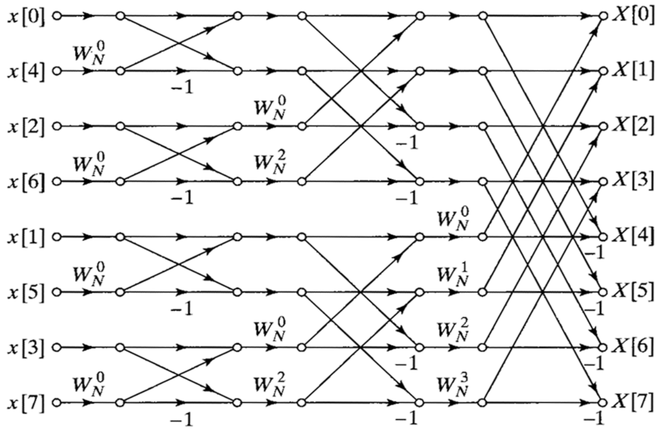
- This suggests that we can modify the FFT algorithm for the inverse DFT computation by the following:
  - Complex conjugate the input DFT coefficients
  - Compute the *forward* FFT
  - Complex conjugate the output of the FFT and multiply by $1/N$
- This method has the advantage that the internal FFT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

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 …..

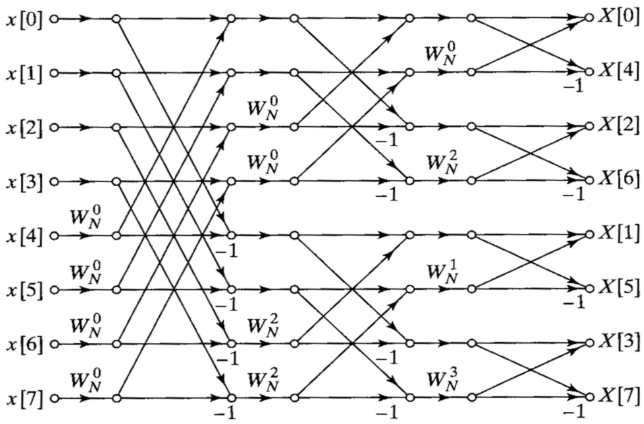
Alternate DIT FFT structures (continued)

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



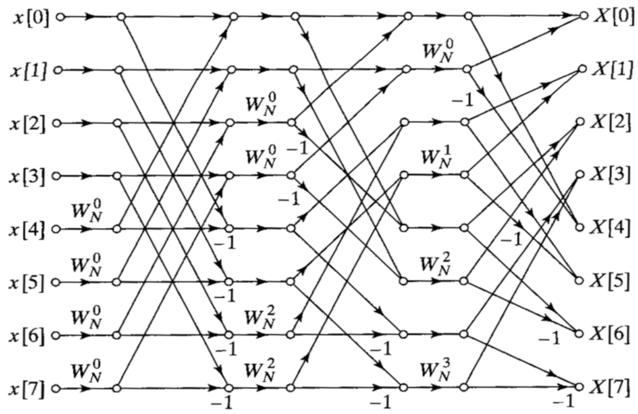
Alternate DIT FFT structures (continued)

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



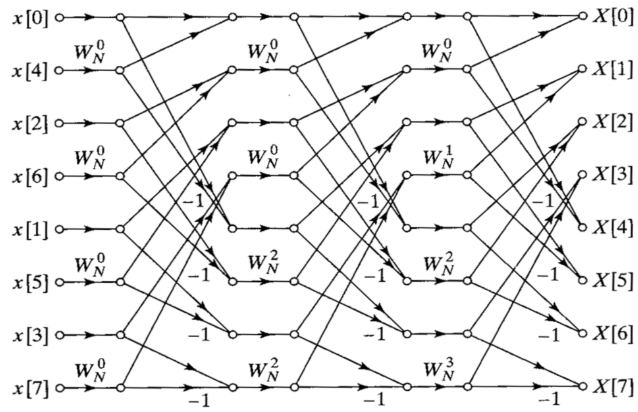
Alternate DIT FFT structures (continued)

- 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

The decimation-in-frequency (DIF) FFT algorithm

- 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:

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

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

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

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

Continuing with decimation in frequency ...

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

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

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

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

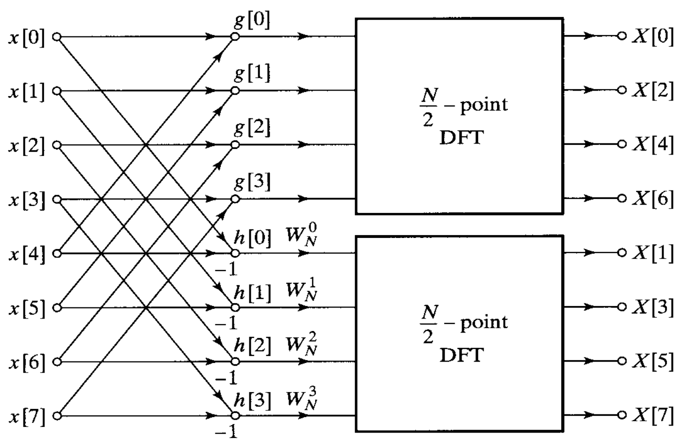
$$X[k]=\sum_{n=0}^{(N/2)-1}\Big[x[n]+(-1)^{2r}(-1)x[n+(N/2)]\Big]W_N^{n(2r+1)}$$

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

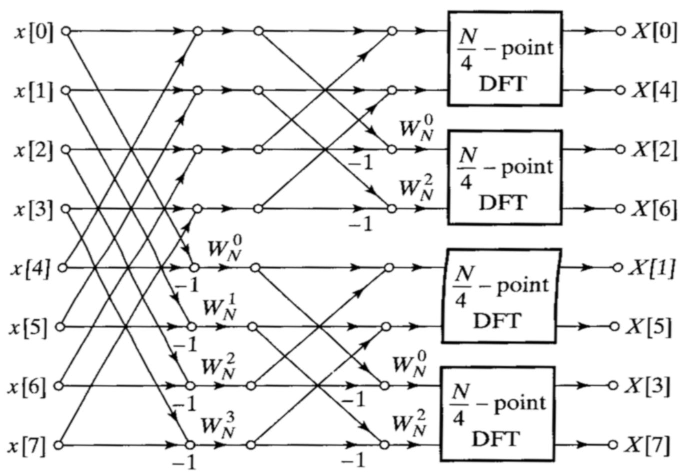
- These expressions are the *N/2*-point DFTs of

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

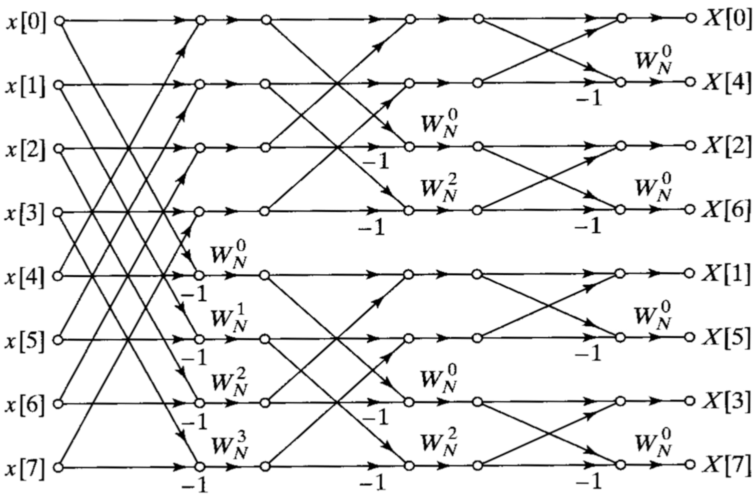
These equations describe the following structure:



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

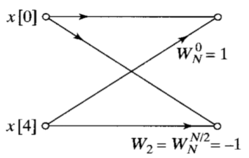


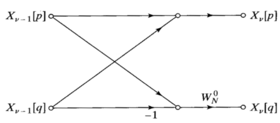
… and replacing the *N/4*-point DFTs by butterflys we obtain



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:



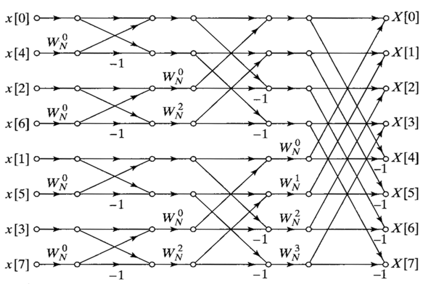


- Comment:
  - We will revisit transposed forms again in our discussion of filter implementation

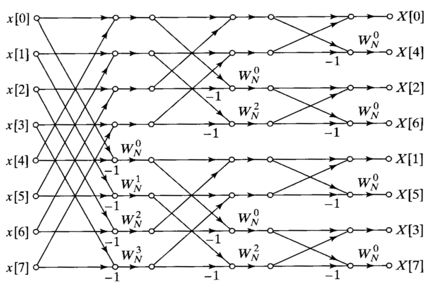
The DIF FFT is the transpose of the DIT FFT

Comparing DIT and DIF structures:

DIT FFT structure:



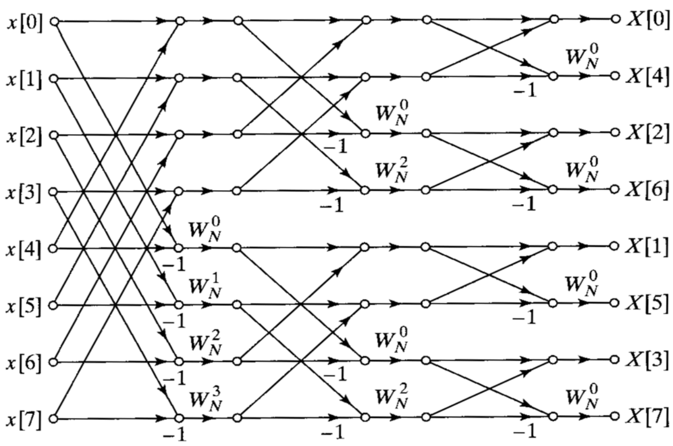
DIF FFT structure:



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

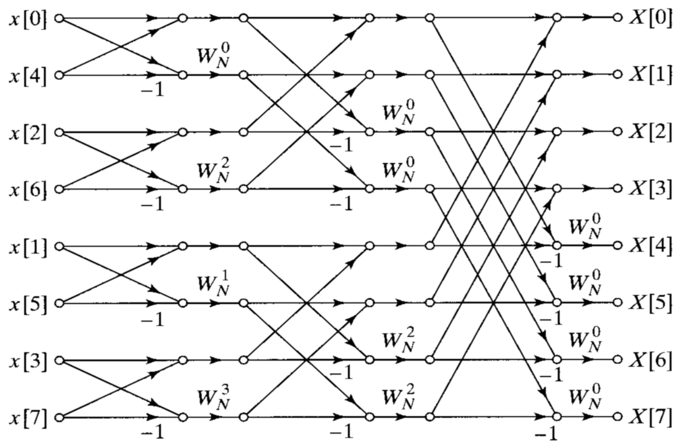
Alternate DIF FFT structures

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



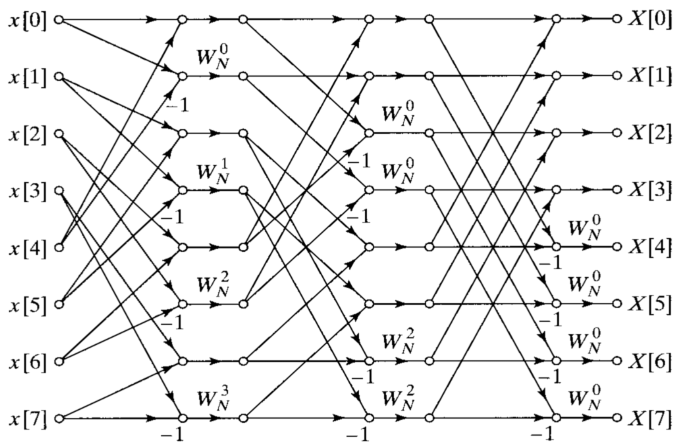
Alternate DIF FFT structures (continued)

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



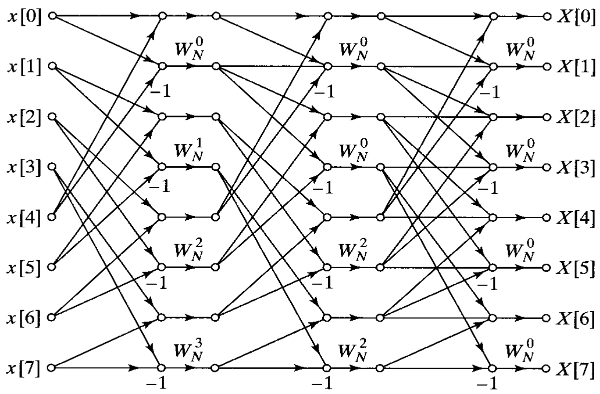
Alternate DIF FFT structures (continued)

- 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 …

Let $N = p_1 p_2 p_3 p_4 ... p_v$; $q_1 = N / p_1$, $q_2 = N / p_1 p_2$, 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, 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

Summary

- We have 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