Introduction to Fast Fourier Transform (FFT) Algorithms

R.C. Maher

EE477 DSP

Spring 2007

Discrete Fourier Transform (DFT)

- The DFT provides uniformly spaced samples of the Discrete-Time Fourier Transform (DTFT)
- DFT definition:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi nk}{N}} \qquad\qquad x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[k] e^{j\frac{2\pi nk}{N}}$$

- Requires N^2 complex multiplies and $N(N-1)$ complex additions

Faster DFT computation?

- Take advantage of the symmetry and periodicity of the complex exponential:
  - symmetry: $e^{-j2\pi k[N-n]/N} = e^{+j2\pi kn/N} = \left(e^{-j2\pi kn/N}\right)^*$
  - periodicity: $e^{-j2\pi nk/N} = e^{-j2\pi[n+N]k/N} = e^{-j2\pi n[k+N]/N}$

- Note that two length $N/2$ DFTs take less computation than one length N DFT: $2(N/2)^2 < N^2$

- Algorithms that exploit computational savings are collectively called *Fast Fourier Transforms*

Decimation-in-Time Algorithm

- Consider expressing DFT with even and odd input samples:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk/N}$$

$$= \sum_{n\ even} x[n] e^{-j2\pi nk/N} + \sum_{n\ odd} x[n] e^{-j2\pi nk/N}$$

$$= \sum_{r=0}^{\frac{N}{2}-1} x[2r] (e^{-j4\pi/N})^{rk} + e^{-j2\pi k/N} \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] (e^{-j4\pi/N})^{rk}$$

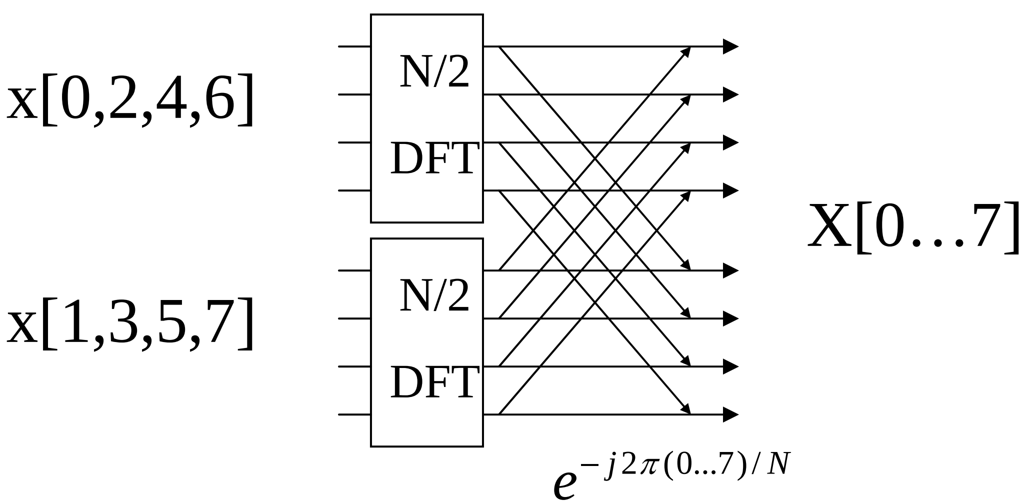
$$= \sum_{r=0}^{\frac{N}{2}-1} x[2r] e^{-j2\pi rk/\left(N/2\right)} + e^{-j2\pi k/N} \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] e^{-j2\pi rk/\left(N/2\right)}$$

DIT Algorithm (cont.)

- Result is the sum of two N/2 length DFTs

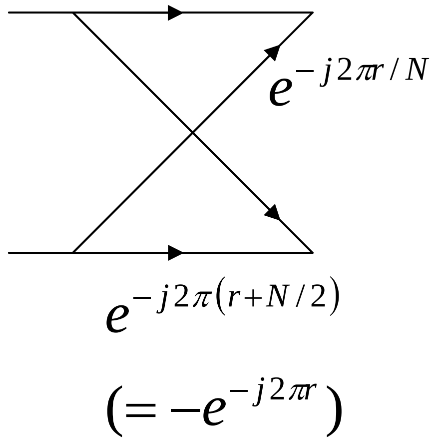
$$X[k] = \underbrace{G[k]}_{\substack{\text{N/2 DFT} \\ \text{of even samples}}} + e^{-j2\pi k/N} \cdot \underbrace{H[k]}_{\substack{\text{N/2 DFT} \\ \text{of odd samples}}}$$

- Then repeat decomposition of N/2 to N/4 DFTs, etc.

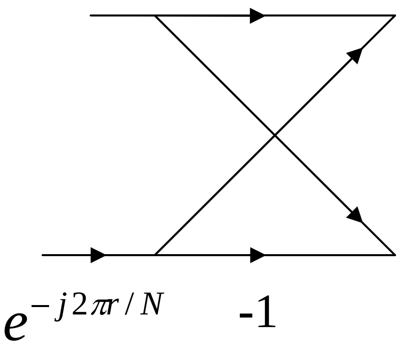


Detail of “Butterfly”

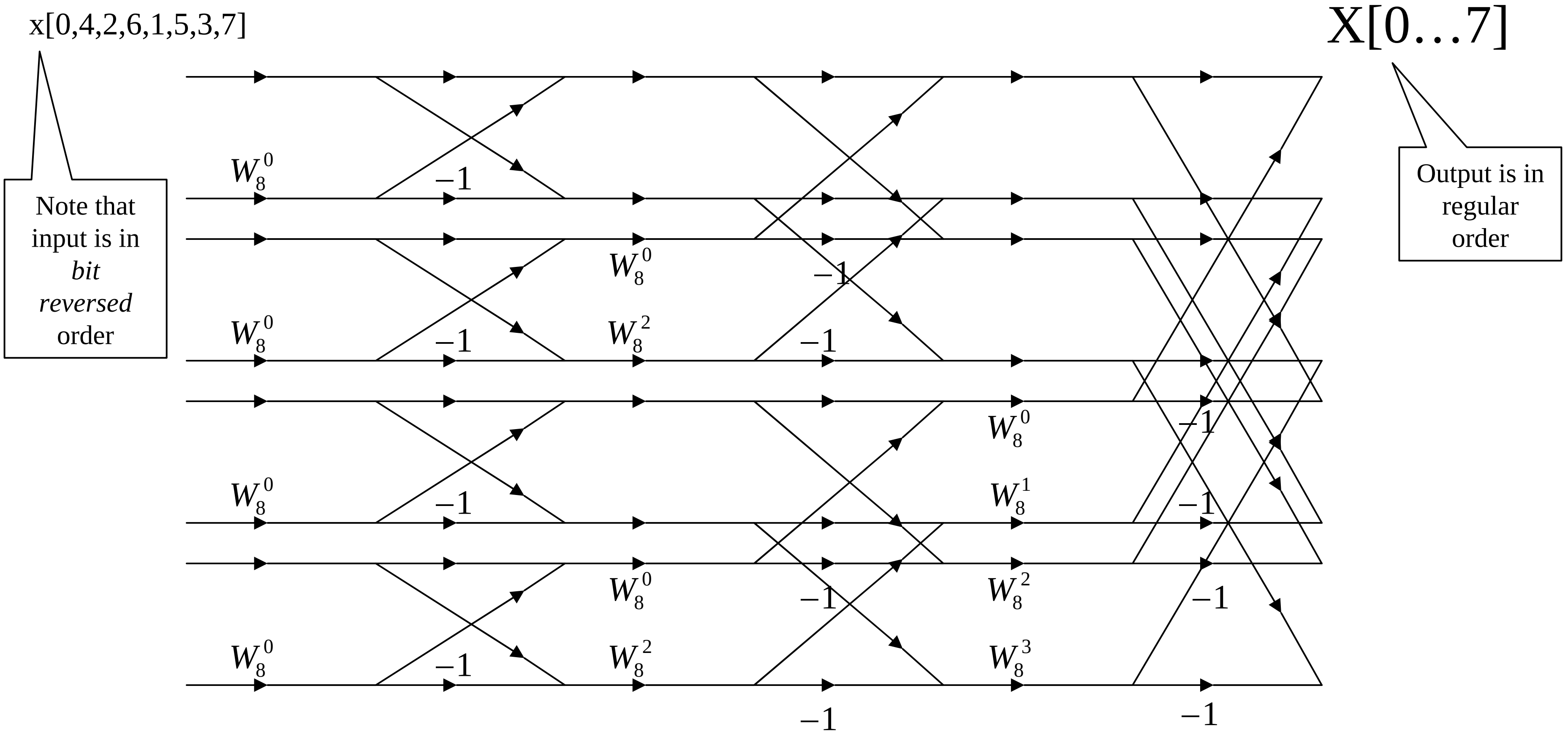
- Cross feed of G[k] and H[k] in flow diagram is called a “butterfly”, due to shape



or simplify:



8-point DFT Diagram



$$W_8^0 = 1; \quad W_8^1 = e^{-j\pi/4}; \quad W_8^2 = e^{-j\pi/2} = -j; \quad W_8^3 = e^{-j3\pi/4}; \quad W_8^4 = e^{-j\pi} = -1$$

Computation on DSP

- Input and Output data
  - Real data in X memory
  - Imaginary data in Y memory
- Coefficients ("twiddle" factors)
  - `cos(real)` values in X memory
  - `sin(imag)` values in Y memory
- Inverse computed with exponent sign change and 1/N scaling