CMPSCI 691AD – General Purpose Computation on the GPU

Spring 2009

Lecture 17: Fast Fourier Transform

Rui Wang

Topics

- Discrete Fourier Transform (DFT)
- Fast Fourier Transform (FFT)
- Applications of FFT
- CUFFT, DCT8x8

Discrete Fourier Transform

- Fourier Transform: represent a signal (function) as a linear sum of sines/cosines.

Discrete Fourier Transform

- Fourier Transform: represent a signal (function) as a linear sum of sines/cosines.
- Discrete Fourier Transform:
  - N input real/complex numbers → N output, complex numbers

$$X_k = \sum_{n=0}^{N-1} x_n \, e^{-\frac{2\pi i}{N} k n} \qquad k = 0, 1, 2, \ldots, N-1$$

$$i = \sqrt{-1}$$

Discrete Fourier Transform

- Fourier Transform: represent a signal (function) as a linear sum of sines/cosines.
- Discrete Fourier Transform:
  - N input real/complex numbers → N output, complex numbers

$$X_k = \sum_{n=0}^{N-1} x_n \, \omega^{k\,n} \qquad k = 0, 1, 2, \ldots, N-1$$

$$\omega = e^{-\frac{2\pi i}{N}}$$

Nth root of unity

Discrete Fourier Transform

- Discrete Fourier Transform:
  - N input real/complex numbers → N output, complex numbers
  - We can think of this as a basis projection
    - Orthogonal basis set, energy conservation

$$X_k = \sum_{n=0}^{N-1} x_n \, \omega^{k\,n} \qquad k = 0, 1, 2, \ldots, N-1$$

$$\omega = e^{-\frac{2\pi i}{N}}$$

Nth root of unity

Discrete Fourier Transform

- Inverse Discrete Fourier Transform:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \, e^{\frac{2\pi i}{N} k n} \qquad n = 0, 1, 2, \ldots, N-1$$

Fast Fourier Transform

- Direct Computation: O(N^2)
- Fast Fourier Transform (FFT): O(N logN)
  - Many different algorithms
  - We will focus on Cooley-Tukey algorithm
    - Basically a divide and conquer algorithm

Fast Fourier Transform

- Repeatedly divide an N point DFT to two N/2 FFTs:

$$X_k = \sum_{n=0}^{N-1} x_n \, \omega^{k n}$$

Fast Fourier Transform

- Repeatedly divide an N point DFT to two N/2 FFTs:

$$X_k = \sum_{n=0}^{N-1} x_n \, \omega^{k n}$$

$$= \sum_{m=0}^{N/2-1} x_{2m} \, \omega^{2mk} + \sum_{m=0}^{N/2-1} x_{2m+1} \, \omega^{(2m+1)k}$$

Fast Fourier Transform

- Repeatedly divide an N point DFT to two N/2 FFTs:

$$X_k = \sum_{n=0}^{N-1} x_n \, \omega^{kn}$$

$$= \sum_{m=0}^{N/2-1} x_{2m} \, \omega^{2mk} + \sum_{m=0}^{N/2-1} x_{2m+1} \, \omega^{(2m+1)k}$$

$$= \underline{\sum_{m=0}^{N/2-1} x_{2m} \, (\omega^2)^{km}} + \omega^k \underline{\sum_{m=0}^{N/2-1} x_{2m+1} \, (\omega^2)^{km}}$$

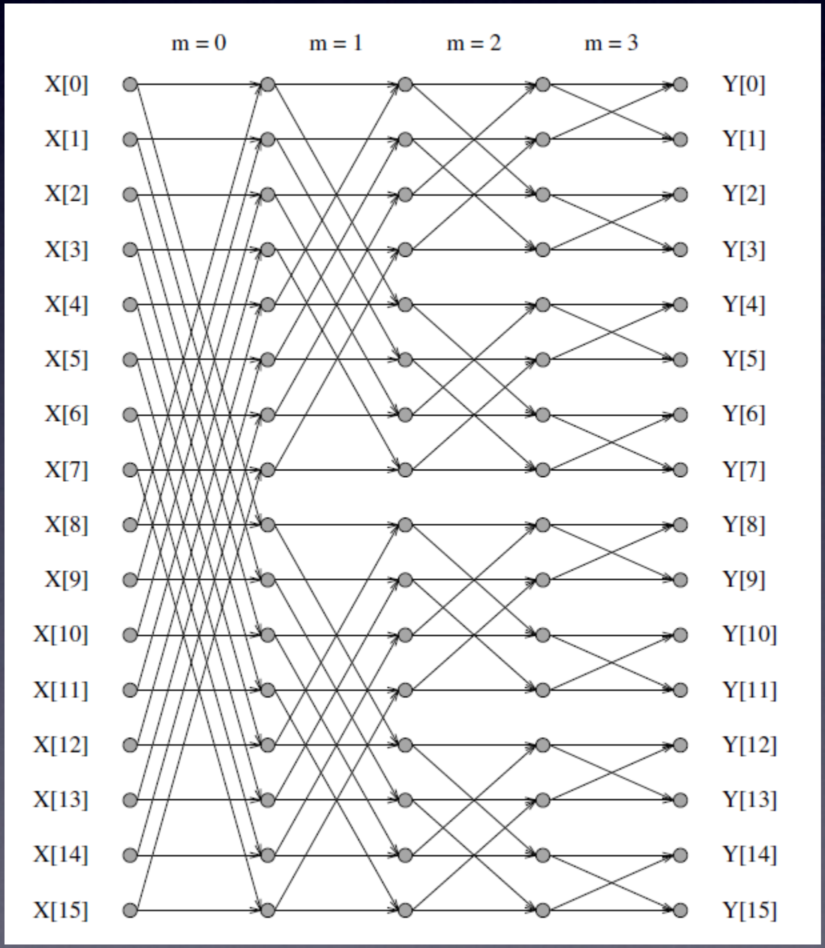
Fast Fourier Transform

- Sequential algorithm: (recursive)

procedure R_FFT(X, Y, n, ω)
if $(n = 1)$ **then** $Y[0] := X[0]$ **else**
begin
    R_FFT($\langle X[0], X[2], \ldots, X[n-2] \rangle, \langle Q[0], Q[1], \ldots, Q[n/2] \rangle, n/2, \omega^2$);
    R_FFT($\langle X[1], X[3], \ldots, X[n-1] \rangle, \langle T[0], T[1], \ldots, T[n/2] \rangle, n/2, \omega^2$);
    for $i := 0$ **to** $n-1$ **do**
        $Y[i] := Q[i \bmod (n/2)] + \omega^i T[i \bmod (n/2)]$;
end R_FFT

Fast Fourier Transform

- Complexity: O(N log N)
- Parallel Implementation: similar to a sorting network



Fast Fourier Transform

- Applications of FFT:
  - Spectral Analysis
  - Data Compression
  - Partial Differential Equation (PDE)
  - Large-scale convolution
    - Convolution Theorem

Fast Fourier Transform

- CUFFT
- DCT8x8 (CUDA SDK)