Final Project

- Freescale DSP56300 Family Manuals collected this week.
- Project #4 is due on Friday, Apr 17.
- Online course evaluations begin April 20. Course evaluations are required—your feedback is very important!

Discrete-Time Fourier Transform (DTFT)

The DTFT is a transformation (mapping) from a discrete-time sequence, $x[n]$ to a continuous-frequency function, $X(e^{j\omega})$. The synthesis and analysis formulae are given below

$$x[n] = \frac{1}{2\pi}\int_{2\pi} X(e^{j\omega})e^{j\omega n}d\omega \leftrightarrow X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}.$$

Discrete Fourier Transform (DFT)

Consider the N-point DFT of a length L signal

$$X(\omega_k) = \sum_{n=0}^{L-1} x[n]e^{-j\omega_k n}$$

where

$$\omega_k = 2\pi k/N.$$

is the kth DFT frequency for $0 \le k \le N-1$. Substitution yields

$$X(\omega_k) = \sum_{n=0}^{L-1} x[n]e^{-j2\pi kn/N}$$

For convenience let the Nth root of unity (often called a twiddle factor) be denoted as

$$W_N = e^{-j2\pi/N}.$$

Substituting, the DFT analysis formula can be rewritten as (replacing ω_k simply with the index k)

$$X[k] = \sum_{n=0}^{L-1} x[n]W_N^{kn},\ 0 \le k \le N-1.$$

The synthesis formula is

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

For a finite-length signal, it follows that the DFT is a *sampled* form of the DTFT

$$X[k] = \sum_{n=0}^{L-1} x[n]e^{-j(2\pi/N)kn} = \sum_{n=-\infty}^{\infty} x[n]e^{-j(2\pi/N)kn} = X(e^{j\omega})\Big|_{\omega=2\pi k/N}$$

Fast Fourier Transform

The basic idea in the FFT is to decompose the DFT into smaller transforms, perform these smaller transforms and combine the results, i.e. "divide and conquer". The decomposition exploits symmetries in the DFT matrix.

We first assume that the length of the signal $L = N$ where N is the number of evaluation points in the DFT. We can achieve $L = N$ by zero padding or modulo N wrapping. We therefore have

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

We'll also assume that N is a power of 2. We usually have enough flexibility to do this.

Decimation-in-Time FFT

We begin by first partitioning the DFT sum above into two sums: one taken over the even indices and the other taken over the odd indices

$$\begin{aligned} X[k] &= \sum_{n=0}^{N-1} x[n] W_N^{kn}, \quad k = 0, 1, \text{K}, N-1 \\ &= \sum_{n=\text{even}}^{N-1} x[n] W_N^{kn} + \sum_{n=\text{odd}}^{N-1} x[n] W_N^{kn} \\ &= \sum_{r=0}^{\frac{N}{2}-1} x[2r] W_N^{k(2r)} + \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] W_N^{k(2r+1)} \end{aligned} \tag{1}$$

Next we note that

$$\begin{aligned} W_N^{k(2r)} &= e^{-j\frac{2\pi}{N}k(2r)} = e^{-j\frac{2\pi}{N/2}kr} = W_{N/2}^{kr} \\ W_N^{k(2r+1)} &= e^{-j\frac{2\pi}{N}k(2r+1)} = e^{-j\frac{2\pi}{N/2}kr} e^{-j\frac{2\pi}{N}k} = W_N^k W_{N/2}^{kr} \end{aligned}. \tag{2}$$

Substituting (2) into (1) we have

$$\begin{aligned} X[k] &= \sum_{r=0}^{\frac{N}{2}-1} W_{N/2}^{kr} x[2r] + W_N^k \sum_{r=0}^{\frac{N}{2}-1} W_{N/2}^{kr} x[2r+1], \quad k = 0, 1, \text{L}, N-1 \\ &= G[k] + W_N^k H[k] \end{aligned}. \tag{3}$$

where $G[k]$, $H[k]$ are the $N / 2$ point DFTs of the length $N / 2$ sequences $x[2r]$, $x[2r + 1]$ respectively

$$G[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r] W_{N/2}^{kr}, \quad k = 0, 1, \text{L}, \frac{N}{2}-1$$

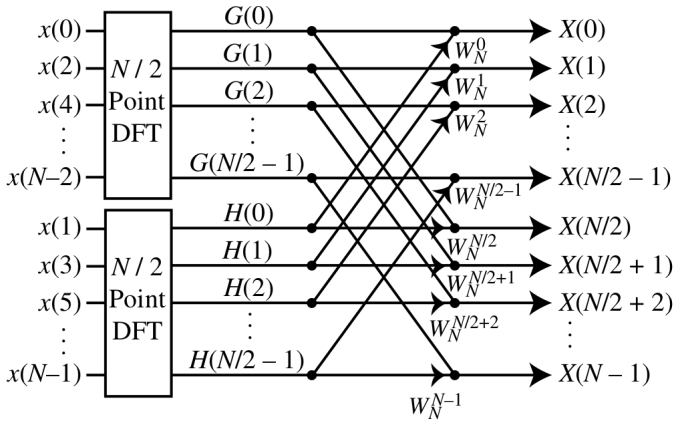
$$H[k] = \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] W_{N/2}^{kr}, \quad k = 0, 1, \text{L}, \frac{N}{2}-1$$

Note that $G[k]$, $H[k]$ are periodic with a period of $N / 2$, i.e.

$$\begin{aligned} G[k + N/2] &= \sum_{r=0}^{N/2-1} x[2r] W_{N/2}^{(k+N/2)r} \\ &= \sum_{r=0}^{N/2-1} x[2r] W_{N/2}^{kr} \\ &= G[k] \end{aligned} \tag{4}$$

We see from (3) that the N-point DFT can be computed with a pair of $N / 2$ point DFTs as defined and by exploiting the

fact that $G[k]$, $H[k]$ are periodic with a period of $N / 2$. The process of DFTs and recombination can be seen graphically below



We see at this point the computational complexity for an N-point DFT using this approach is

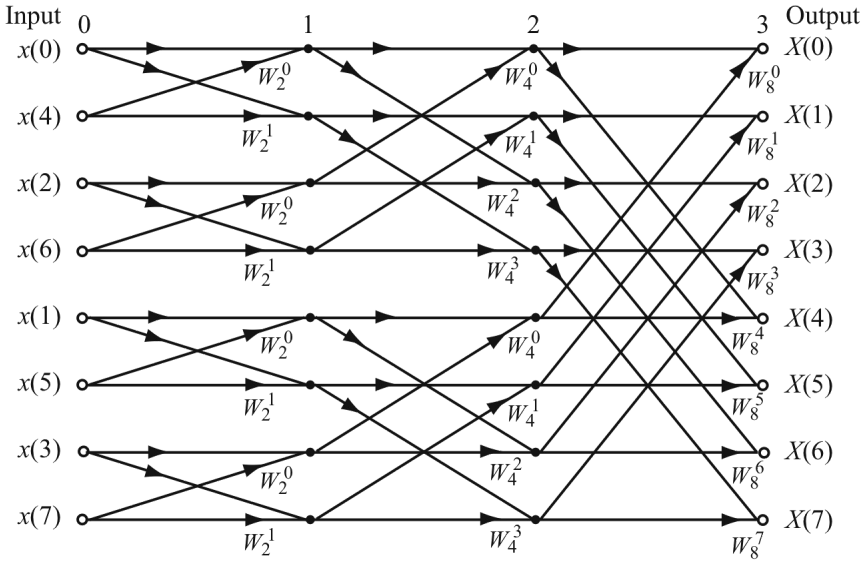
(2) $N / 2$-point DFTs each requiring $\left(\frac{N}{2}\right)^2$ complex MACs

(N) complex MACs for recombination necessary to calculate final $X[k]$s

This yields $\frac{N^2}{2} + N$ complex MACs versus N^2 complex MACs for the conventional DFT calculation.

Example: Let $N = 1024$. For a DFT implemented in conventional form, we require 1, 048, 576 complex MACs while that implemented with a divide and conquer approach requires 525, 312 complex MACs. The latter approach reduces computation by about 50%.

As in this first stage, we can decompose (or decimate) each $N / 2$-point DFT into two $N / 4$-point DFTs and another recombination scheme. This can be continued until the problem has been decomposed down to a series of 2-point DFTs (a total of $\log_2(N)$ stages) and a bunch of recombination schemes as illustrated below.

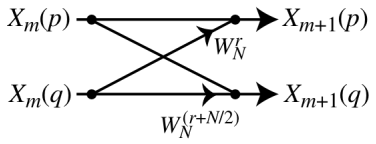


We notice several things in this example and from the figure:

1) In order to get the $X[k]$s in order, we must scramble the $x[n]$s. The scrambling scheme is determined as follows. The $x[n]$ required for position m is determined by representing m in binary form. The bits representing m are reversed and n is obtained by converting these bits back to decimal form. Therefore, $x[n]$ must first be placed in bit-reversed order prior to FFT. On a DSP, bit-reversed addressing modes are typically done in hardware.

Input Index	Binary Input Index	Binary Output Index (Bit Reversed)	Output Index
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

2) The basic computational block in the FFT is called the "butterfly"



where the I/O equations are given by

$$X_{m+1}[p] = X_m[p] + W_N^r X_m[q]$$
$$X_{m+1}[q] = X_m[p] + W_N^{r+N/2} X_m[q]$$

Note that the butterfly outputs $X_{m+1}(p),\ X_{m+1}(q)$ depend only on the inputs to the butterfly $X_m(p),\ X_m(q)$. Therefore memory locations storing $X_m(p),\ X_m(q)$ are easily updated with $X_{m+1}(p),\ X_{m+1}(q)$. This kind of computation is referred to as an "in-place" calculation. Fast memory reads/writes (or even dual reads/writes in one clock cycle) and MACs are essential to fast Butterfly computations and are also optimized in VLSI.

It can be shown that the Decimation-in-Time FFT requires $\frac{N}{2}\log_2(N)$ complex multiplications. Order $N\log(N)$, denoted $O(N\log N)$. algorithms are often called "fast" hence the name Fast Fourier Transform.

Example If we compute the DFT of a 1024-point sequence, we require 1,048,576 complex MACs using the conventional DFT and 5120 complex MACs using the FFT. This relates to over a 200-fold reduction computation!