1.6.4 FAST FOURIER TRANSFORM (FFT) ALGORITHM

The FFT is an *algorithm* for efficient computation of the DFT.

It is *not* a new transform.

Recall

$$X^{(N)}(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi kn/N}, \quad k = 0,1,...,N-1$$

Superscript (N) is to show length of DFT. For each value of k, computation of X(k) requires:

N	complex multiplications
N−1	complex additions

Define a *complex operation* (CO) as 1 complex multiplication and 1 complex addition.

Computation of length N DFT then requires approximately N^2 CO's.

To derive an efficient algorithm for computation of the DFT, we employ a divide-and-conquer strategy.

Assume N is even.

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

$$X^{(N)}(k) = \sum_{\substack{n=0 \\ n \text{ even}}}^{N-1} x(n)e^{-j2\pi kn/N}$$

$$+ \sum_{\substack{n=0 \\ n \text{ odd}}}^{N-1} x(n)e^{-j2\pi kn/N}$$

$$= \sum_{m=0}^{N/2-1} x(2m)e^{-j2\pi k(2m)/N}$$

$$+ \sum_{m=0}^{N/2-1} x(2m+1)e^{-j2\pi k(2m+1)/N}$$

$$X^{(N)}(k) = \sum_{m=0}^{N/2-1} x(2m)e^{-j2\pi km/(N/2)}$$

$$+ e^{-j2\pi k/N} \sum_{m=0}^{N/2-1} x(2m+1)e^{-j2\pi km/(N/2)}$$

Let $x_0(n) = x(2m), \qquad m = 0,\ldots,N/2-1$

$x_1(n) = x(2m+1), \qquad m = 0,\ldots,N/2-1$

Now have

$$X^{(N)}(k) = X_0^{(N/2)}(k) + e^{-j2\pi k/N} X_1^{(N/2)}(k) ,$$

$$k = 0,...,N-1$$

Note that $X_0^{(N/2)}(k)$ and $X_1^{(N/2)}(k)$ are both periodic with period $N/2$, while $e^{-j2\pi k/N}$ is periodic with period N.

Computation

1. Direct

$$X^{(N)}(k)\,, \quad k = 0,...,N-1 \qquad N^2 \text{ CO's}$$

2. Decimation by factor of 2

$$X_0^{(N/2)}(k)\,, \quad k = 0,...,N/2-1 \qquad N^2/4 \text{ CO's}$$
$$X_1^{(N/2)}(k)\,, \quad k = 0,...,N/2-1 \qquad N^2/4 \text{ CO's}$$

$$X^{(N)}(k) = X_0^{(N/2)}(k) + e^{-j2\pi k/N}\, X_1^{(N/2)}(k)\,,$$

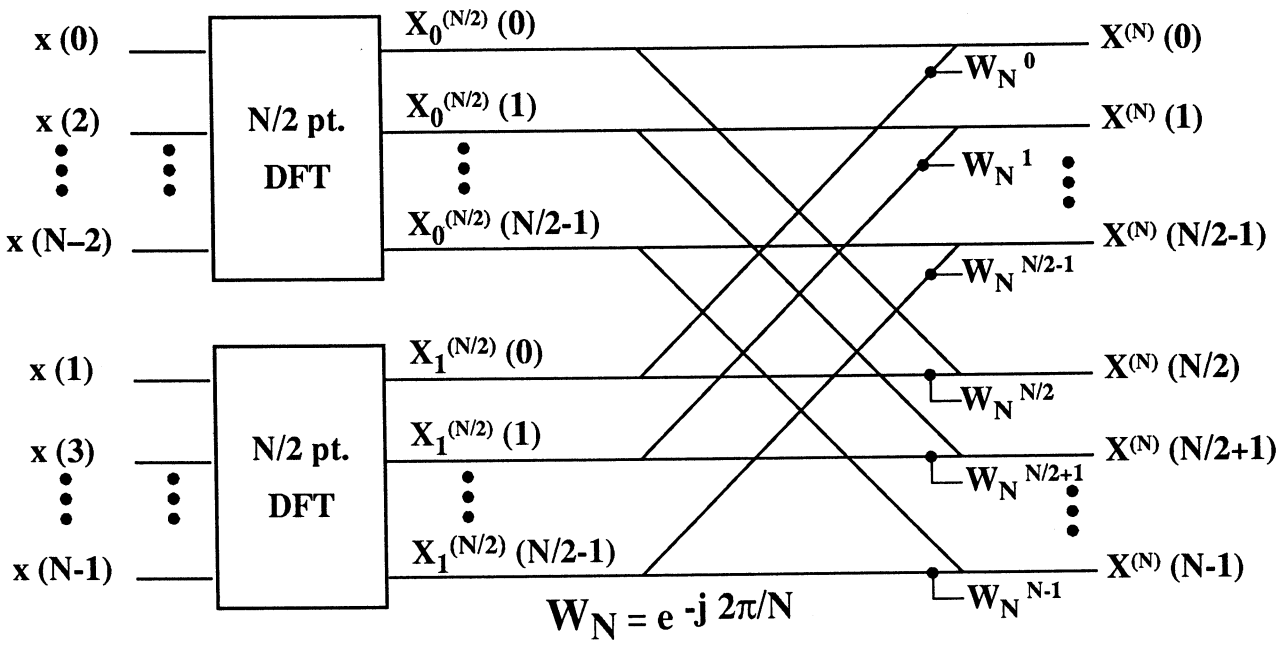
$$k = 0,\ldots,N-1$$

$$N \text{ CO's}$$

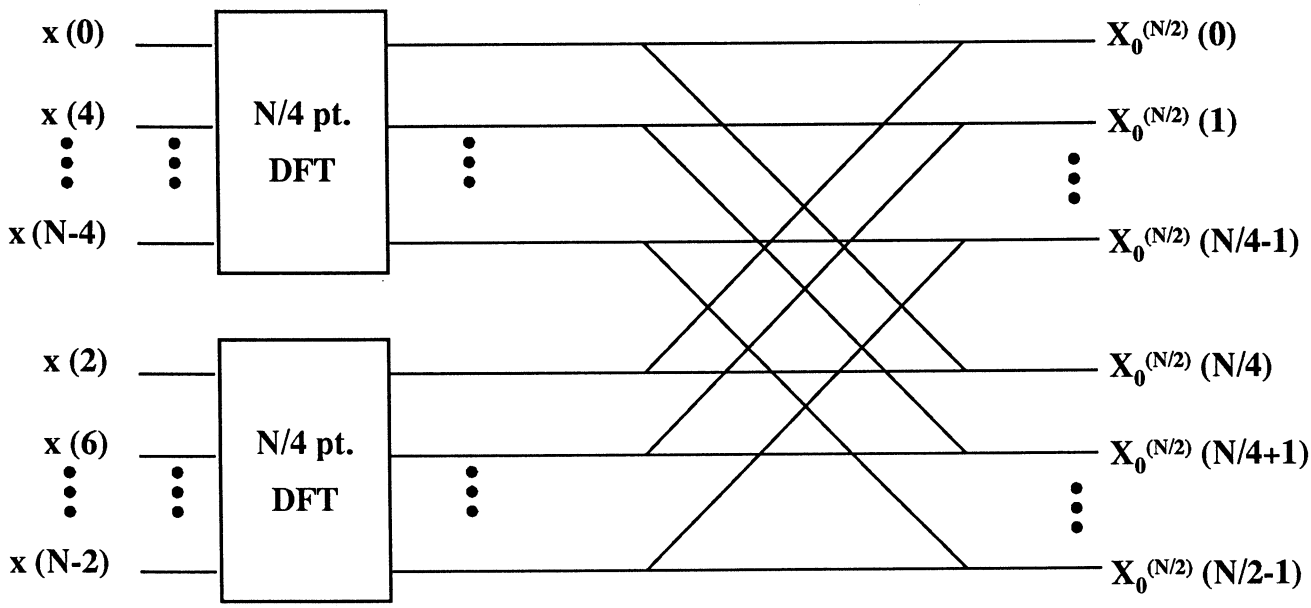
Total $N^2/2 + N$ CO's

For large N, we have nearly halved computation.

Consider a signal flow diagram of what we have done so far:



If N is even, we can repeat the idea with each N/2 pt. DFT:



If $N = 2^M$, we repeat the process M times resulting in M stages.

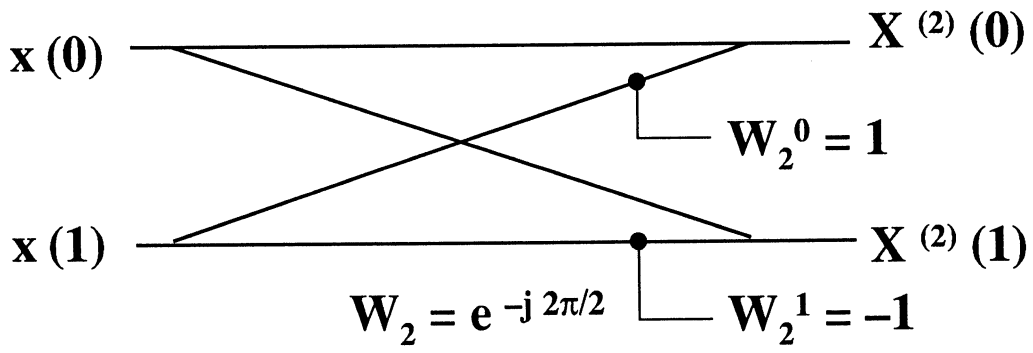
The first stage consists of 2 pt. DFT's

$$X^{(2)}(k) = \sum_{n=0}^{1} x(n)\, e^{-j2\pi kn/2}$$

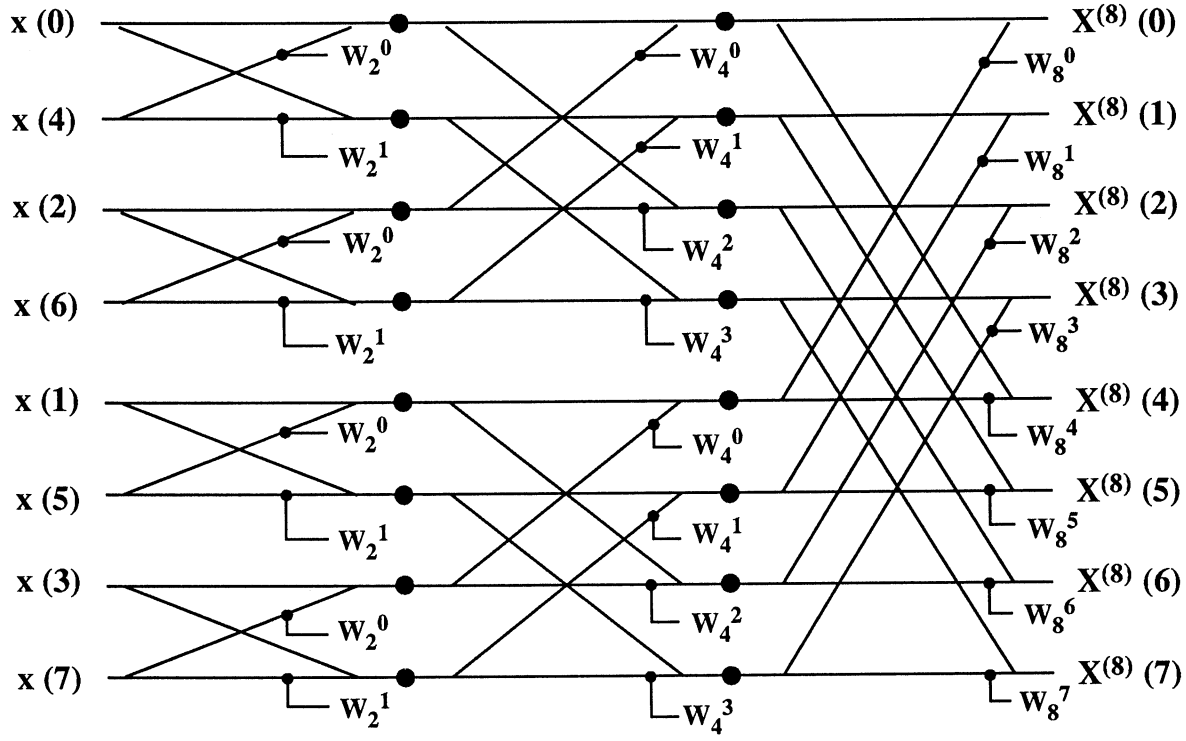
$$X^{(2)}(0) = x(0) + x(1)$$

$$X^{(2)}(1) = x(0) - x(1)$$

Flow diagram of 2 pt. DFT



Full Example for N = 8 (M = 3)



Ordering of Input Data

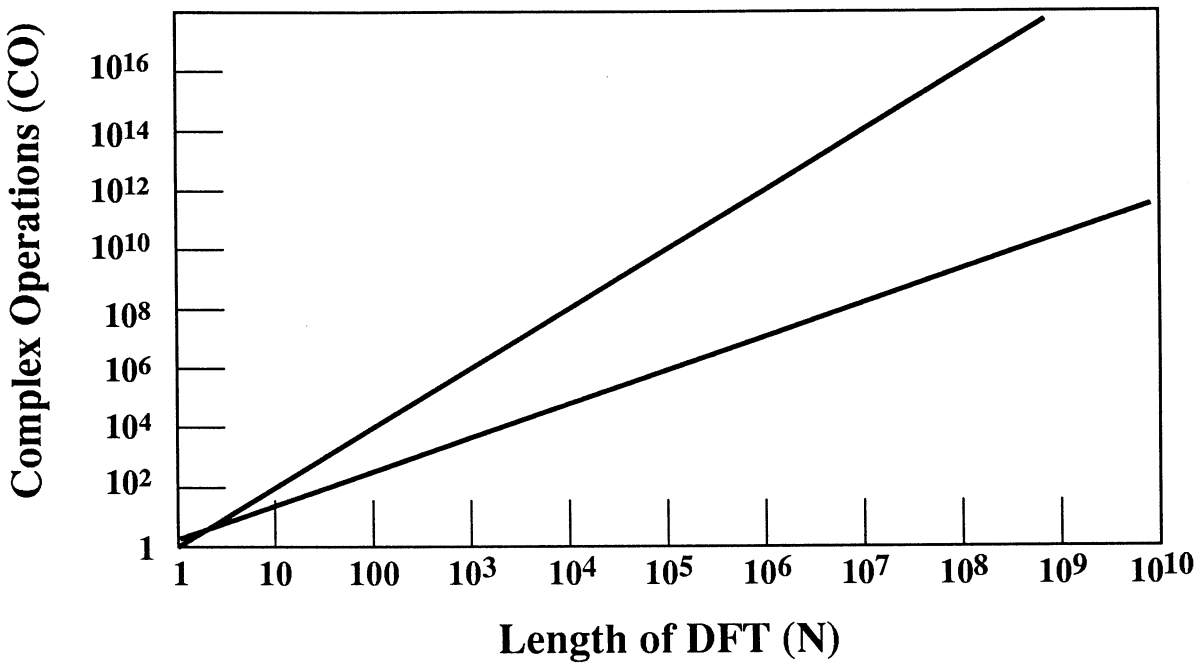
Normal Order		Bit Reversed Order	
Decimal	Binary	Binary	Decimal
0	0 0 0	0 0 0	0
1	0 0 1	1 0 0	4
2	0 1 0	0 1 0	2
3	0 1 1	1 1 0	6
4	1 0 0	0 0 1	1
5	1 0 1	1 0 1	5
6	1 1 0	0 1 1	3
7	1 1 1	1 1 1	7

Computation $(N = 2^M)$

$M = \log_2 N$ stages

N CO′s/stage

Total: $N \log_2 N$ CO′s


Complex Operations (CO)
10^{16}
10^{14}
10^{12}
10^{10}
10^{8}
10^{6}
10^{4}
10^{2}
1
1
10
100
10^{3}
10^{4}
10^{5}
10^{6}
10^{7}
10^{8}
10^{9}
10^{10}
Length of DFT (N)

Comments

- The algorithm we derived is the decimation-in-time radix 2 FFT.
  - input in bit-reversed order
  - in-place computation
  - output in normal order
- The dual of it is the decimation-in-frequency radix 2 FFT.
  - input in normal order
  - in-place computation
  - output in normal order

- The same approaches may be used to derive either decimation-in-time or decimation-in-frequency mixed radix FFT algorithms for any N which is a composite number.

- Another class of FFT algorithms is based on fast techniques for performing small convolutions (Winograd).

- All FFT algorithms are based on composite N and require O(N log N) computation.

- The DFT of a length N real signal can be expressed in terms of a length N/2 DFT.