Discrete Fourier Transform

- Discrete Fourier transform (DFT) pairs

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

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

where \( W_N^{-kn} = e^{-j\frac{2\pi}{N}kn} \)

- DFT/IDFT can be implemented by using the same hardware
- It requires \( N^2 \) complex multiplications and \( N(N-1) \) complex additions
Decimation in Time

\[ X_N[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \quad \leftarrow \text{N–point DFT} \]

\[ = \sum_{n \text{ even}}^{N/2-1} x[2l] W_N^{2lk} + \sum_{n \text{ odd}}^{N/2-1} x[2l+1] W_N^{(2l+1)k} \]

\[ = \sum_{l=0}^{N/2-1} x[2l] \left[ \frac{W_N^2}{W_{N/2}} \right]^{lk} + \sum_{l=0}^{N/2-1} x[2l+1] \left[ \frac{W_N^2}{W_{N/2}} \right]^{lk} \]

\[ = \sum_{l=0}^{N/2-1} x[2l] W_{N/2}^{lk} + W_N^k \sum_{l=0}^{N/2-1} x[2l+1] W_{N/2}^{lk} \]

\[ = \sum_{l=0}^{N/2-1} x[2l] W_{N/2}^{lk} + W_N^k \sum_{l=0}^{N/2-1} x[2l+1] W_{N/2}^{lk} \]

\[ \text{two } N/2–\text{point DFT’s!!!} \]

\[ N+2(N/2)^2 \text{ complex multiplications vs. } N^2 \text{ complex multiplication} \]
Using a briefer system of notation:

\[ X_N[k] = G_{N/2}[k] + W_N^k H_{N/2}[k], \]

where \( G_{N/2}[k] \) and \( H_{N/2}[k] \) are the \( N/2 \)-point DFTs involving \( x[n] \) with even and odd \( n \), respectively.
Flow Graph of the DIT FFT
Corollary:
Any $N$-point DFT with even $N$ can be computed via two $N/2$-point DFTs. In turn, if $N/2$ is even then each of these $N/2$-point DFTs can be computed via two $N/4$-point DFTs and so on. In the case of $N = 2^r$, all $N$, $N/2$, $N/4$ ... are even and such a process of “splitting” ends up with all 2-point DFTs!
8-point DIT DFT
Remarks

- It requires \( v = \log_2 N \) stages
- Each stage has \( N \) complex multiplications and \( N \) complex additions
- The number of complex multiplications (as well as additions) is equal to \( N \log_2 N \)
- By symmetry property, we have (butterfly operation)

\[
W_N^{r+N/2} = W_N^r W_N^{N/2} = W_N^r e^{-j\pi} = -W_N^{N/2}
\]
8-point FFT

Bit-Reversed order

Normal order

x[0] → \( W_N^0 \) → 1 → \( W_N^0 \) → x[1] → \( W_N^0 \) → 1 → \( W_N^0 \) → x[2] → \( W_N^2 \) → \( W_N^2 \) → \( W_N^0 \) → x[3] → \( W_N^0 \) → 1 → \( W_N^0 \) → x[4] → \( W_N^0 \) → 1 → \( W_N^0 \) → x[5] → \( W_N^0 \) → \( W_N^0 \) → \( W_N^0 \) → x[6] → \( W_N^2 \) → \( W_N^2 \) → \( W_N^0 \) → x[7] → \( W_N^0 \) → 1 → \( W_N^0 \) → x[0]
In-Place Computation

The same register array can be used in each stage

Stage 1

- $X_0[000]$ to $x[0]$
- $X_0[001]$ to $x[4]$
- $X_0[010]$ to $x[2]$
- $X_0[011]$ to $x[6]$
- $X_0[100]$ to $x[1]$
- $X_0[101]$ to $x[5]$
- $X_0[110]$ to $x[3]$
- $X_0[111]$ to $x[7]$

Stage 2

- $X_1[000]$ to $X_0[000]$
- $X_1[001]$ to $X_0[001]$
- $X_1[010]$ to $X_0[010]$
- $X_1[011]$ to $X_0[011]$
- $X_1[100]$ to $X_0[100]$
- $X_1[101]$ to $X_0[101]$
- $X_1[110]$ to $X_0[110]$
- $X_1[111]$ to $X_0[111]$

Stage 3

- $X_2[000]$ to $X_1[000]$
- $X_2[001]$ to $X_1[001]$
- $X_2[010]$ to $X_1[010]$
- $X_2[011]$ to $X_1[011]$
- $X_2[100]$ to $X_1[100]$
- $X_2[101]$ to $X_1[101]$
- $X_2[110]$ to $X_1[110]$
- $X_2[111]$ to $X_1[111]$

- $X_3[000]$ to $X_2[000]$
- $X_3[001]$ to $X_2[001]$
- $X_3[010]$ to $X_2[010]$
- $X_3[011]$ to $X_2[011]$
- $X_3[100]$ to $X_2[100]$
- $X_3[101]$ to $X_2[101]$
- $X_3[110]$ to $X_2[110]$
- $X_3[111]$ to $X_2[111]$

$W_N^0$, $W_N^1$, $W_N^2$, $W_N^3$
8-point FFT

Normal order

Bit-reversed order

The original from given by Cooly & Tukey (DIT FFT)

cwliu@twins.ee.nctu.edu.tw
Normal-Order Sorting v.s. Bit-Reversed Sorting

Normal Order

Bit-reversed Order

cwliu@twins.ee.nctu.edu.tw
DFT v.s. Radix-2 FFT

- DFT: $N^2$ complex multiplications and $N(N-1)$ complex additions
- Recall that each butterfly operation requires one complex multiplication and two complex additions
- FFT: $(N/2) \log_2 N$ multiplications and $N \log_2 N$ complex additions

- **In-place computations**: the input and the output nodes for each butterfly operation are horizontally adjacent ➔ only one storage arrays will be required
Alternative Form

Two complex storage arrays are necessary!!
Alternative Form

Parallel processing:
4 BF units

Sequential processing:
1 BF unit

The same register array can be used
Two register arrays are required
Alternative Form 2

The geometry of each stage is identical

Bit-reversed order

Normal order

cwliu@twins.ee.nctu.edu.tw
Decimation in Frequency (DIF)

- Recall that the DFT is
  \[ X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \quad 0 \leq k \leq N - 1 \]

- DIT FFT algorithm is based on the decomposition of the DFT computations by forming small subsequences in time domain index “n”: \( n = 2^\ell \) or \( n = 2^\ell + 1 \)

- One can consider dividing the output sequence \( X[k] \), in frequency domain, into smaller subsequences: \( k = 2^r \) or \( k = 2^r + 1 \):

\[
\begin{align*}
X[k] &= \begin{cases} 
X[2^r] & 0 \leq r \leq \frac{N}{2} - 1 \\
X[2^r + 1] & \end{cases} \\
X[2^r] &= \sum_{n=0}^{N/2-1} x[n] W_N^{2^r n} + \sum_{n=N/2}^{N-1} x[n] W_N^{2^r n} \\
&= \sum_{n=0}^{N/2-1} (x[n] + x[n + \frac{N}{2}]) W_N^{2^r n} \\
&= \sum_{n=0}^{N/2-1} x[n] W_N^{2^r n} + \sum_{n=0}^{N/2-1} x[n + \frac{N}{2}] W_N^{2^r (n + \frac{N}{2})}
\end{align*}
\]

Substitution of variables

\( W_N^{2^r (n + \frac{N}{2})} = W_N^{2^r n} W_N^{rN} = W_N^{2^r n} \)
DIF FFT Algorithm (1)

\[ 0 \leq r \leq \frac{N}{2} - 1 \]

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

is just N/2-point DFT. Similarly,

\[ X[2r + 1] = \sum_{n=0}^{N/2-1} (x[n] - x[n + \frac{N}{2}]) W_{N}^{n(2r+1)} = \sum_{n=0}^{N/2-1} \{x[n] - x[n + \frac{N}{2}]\} W_{N}^{n} W_{N/2}^{nr} \]
DIF FFT Algorithm (2)

\(v = \log_2 N\) stages, each stage has \(N/2\) butterfly operation. 
\((N/2)\log_2 N\) complex multiplications, \(N\) complex additions

cwliu@twins.ee.nctu.edu.tw
Remarks

• The basic butterfly operations for DIT FFT and DIF FFT respectively are transposed-form pair.

• The I/O values of DIT FFT and DIF FFT are the same
• Applying the transpose transform to each DIT FFT algorithm, one obtains DIF FFT algorithm
Implementation Issues

• Radix-2, Radix-4, Radix-8, Split-Radix, Radix-2², …,
• I/O Indexing
• In-place computation
  - Bit-reversed sorting is necessary
  - Efficient use of memory
  - Random access (not sequential) of memory. An address generator unit is required.
  - Good for cascade form: FFT followed by IFFT (or vice versa)
    • E.g. fast convolution algorithm
• Twiddle factors
  - Look up table
  - CORDIC rotator
Recall... Linear Convolution

\[ x[n] \quad \text{Zero padding} \quad \rightarrow \quad \text{N-point DFT} \quad \rightarrow \quad X[k] \quad 0 \leq k \leq N - 1 \]

\[ h[n] \quad \text{Zero padding} \quad \rightarrow \quad \text{N-point DFT} \quad \rightarrow \quad H[k] \quad 0 \leq k \leq N - 1 \quad N \geq L + P - 1 \]

\[ \times \rightarrow \quad \text{N-point IDFT} \quad \rightarrow \quad y[n] \quad 0 \leq n \leq N + P - 2 \]
Fast Convolution with the FFT

• Given two sequences $x_1$ and $x_2$ of length $N_1$ and $N_2$ respectively
  - Direct implementation requires $N_1 N_2$ complex multiplications
• Consider using FFT to convolve two sequences:
  - Pick $N$, a power of 2, such that $N \geq N_1 + N_2 - 1$
  - Zero-pad $x_1$ and $x_2$ to length $N$
  - Compute $N$-point FFTs of zero-padded $x_1$ and $x_2$, one obtains $X_1$ and $X_2$
  - Multiply $X_1$ and $X_2$
  - Apply the IFFT to obtain the convolution sum of $x_1$ and $x_2$
  - Computation complexity: $2(N/2) \log_2 N + N + (N/2)\log_2 N$
Other Fast Algorithm for DFT

• **Goertzel Algorithm**
  - By reformulating DFT as a convolution
  - It is not restricted to computation of the DFT but any desired set of samples of the Fourier transform of a sequence

• **Winograd Algorithm**
  - An efficient algorithm for computing short convolutions
  - The number of multiplication complexity is of order $O(N)$, however the number of addition complexity is significantly increased.

• **Chirp Transform Algorithm**