Computation of DFT using FFT algorithm

- FFT is a highly efficient procedure for computing the DFT of the finite series & requires less no. of computations than that of direct evaluation of DFT.

\[
X(k) = \sum_{n=0}^{N-1} x(n)W_N^{kn}, 0 \leq k \leq N - 1
\]

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

Twiddle factor \( W_N = e^{-j\frac{2\pi}{N}} \)
Why FFT is needed?

**DFT**
- To evaluate $N$-point DFT
- It requires
  - $N(N+1)$ complex additions
  - $N^2$ complex multiplications

**FFT**
- To evaluate $N$-point DFT
- It requires
  - $N \log_2 N$ complex additions
  - $\frac{N}{2} \log_2 N$ complex multiplications
What is radix-2 FFT?

- If the number of output points $N$ can be expressed as a power of 2, i.e $N = 2^M$, $M$ is an integer.

- Then this algorithm is known as radix-2 FFT algorithm.
Advantages of FFT over DFT

- FFT is the algorithm used to compute DFT fast
- Computationally efficient than direct computation of DFT
- Exploit periodicity and symmetry properties of DFT
- It makes use of the periodicity and symmetry properties of twiddle factor $W_N^k$ to reduce the DFT computation time.
Applications of FFT

- Linear filtering
- Correlation
- Spectrum Analysis

Properties of Twiddle factor:

Symmetry Property

\[ W_{N}^{k+N/2} = -W_{N}^{k} \]

Periodicity Property

\[ W_{N}^{k+N} = W_{N}^{k} \]
Radix-2 DIT FFT algorithm

- The N-point DFT of a sequence $x(n)$ converts the time domain N point sequence $x(n)$ to a frequency domain N-point sequence $X(k)$.
- In DIT the $x(n)$ is decimated and smaller point DFTs are performed.
- The results of smaller point DFTs are combined to get the result of N-point DFT.
- Here, the N-point DFT can be realized from two no. of N/2 point DFTs, the N/2 point DFT can be realized from two no. of N/4 point DFTs and so on.
Radix-2 DIT FFT algorithm  Cont..

• Normal order
  • x(0)-000
  • x(1)-001
  • x(2)-010
  • x(3)-011
  • x(4)-100
  • x(5)-101
  • x(6)-110
  • x(7)-111

Output should be in normal order

• Bit reverse order
  • x(0)-000
  • x(4)-100
  • x(2)-010
  • x(6)-110
  • x(1)-001
  • x(5)-101
  • x(3)-011
  • x(7)-111
Radix-2 DIT FFT algorithm  Cont..

EC8553 Discrete Time Signal Processing
First stage of computation

\[ V_{11}(0) = x(0) + x(4) \]
\[ V_{11}(1) = x(0) - x(4) \]
\[ V_{12}(0) = x(2) + x(6) \]
\[ V_{12}(1) = x(2) - x(6) \]
\[ V_{21}(0) = x(1) + x(5) \]
\[ V_{21}(1) = x(1) - x(5) \]
\[ V_{22}(0) = x(3) + x(7) \]
\[ V_{22}(1) = x(3) - x(7) \]
First stage of computation
Second stage of computation

\[ F_1(0) = V_{11}(0) + W^0_{12}V_{12}(0) \]

\[ F_2(0) = V_{21}(0) + W^0_{22}V_{22}(0) \]

\[ F_1(1) = V_{11}(1) + W^1_{12}V_{12}(1) \]

\[ F_2(1) = V_{21}(1) + W^1_{22}V_{22}(1) \]

\[ F_1(2) = V_{11}(0) - W^0_{12}V_{12}(0) \]

\[ F_2(2) = V_{21}(0) - W^0_{22}V_{22}(0) \]

\[ F_1(3) = V_{11}(1) - W^1_{12}V_{12}(1) \]

\[ F_2(3) = V_{21}(1) - W^1_{22}V_{22}(1) \]
Second stage of computation

\[ V_{11}(0) \rightarrow 1 \rightarrow 1 \rightarrow F_1(0) \]
\[ V_{11}(1) \rightarrow 1 \rightarrow 1 \rightarrow F_1(1) \]
\[ V_{12}(0) \rightarrow 1 \rightarrow 1 \rightarrow F_1(2) \]
\[ V_{12}(1) \rightarrow 1 \rightarrow 1 \rightarrow F_1(3) \]
\[ V_{21}(0) \rightarrow 1 \rightarrow 1 \rightarrow F_2(0) \]
\[ V_{21}(1) \rightarrow 1 \rightarrow 1 \rightarrow F_2(1) \]
\[ V_{22}(0) \rightarrow 1 \rightarrow 1 \rightarrow F_2(2) \]
\[ V_{22}(1) \rightarrow 1 \rightarrow 1 \rightarrow F_2(3) \]
Third stage of computation

\[ X(0) = F(0) + W_0 F(0) \quad X(4) = F(0) - W_0 F(0) \]

\[ X(1) = F(1) + W_1 F(1) \quad X(5) = F(1) - W_1 F(1) \]

\[ X(2) = F(2) + W_2 F(2) \quad X(6) = F(2) - W_2 F(2) \]

\[ X(3) = F(3) + W_3 F(3) \quad X(7) = F(3) - W_3 F(3) \]
Third stage of computation
Butterfly diagram for 8-point radix-2 DIT FFT algorithm
Butterfly diagram for 8-point radix-2 DIT FFT algorithm
Butterfly diagram for 4-point radix-2 DIT FFT algorithm
Calculation of Twiddle factor values

\[ W_N^k = e^{-j2\pi k/N} \]

\[ N \quad k \quad j \quad 2 \quad \pi \quad k \]

\[ W_N^2 = e^{j(-2\pi/2)} = e^{j(-\pi)} = W_{N/2} \]

\[ N \quad 2 \quad j \quad \sin \quad N \quad 2 \]

\[ W_2^0 = W_4^0 = W_8^0 = e^{j(0)} = e^0 = 1 \]

\[ j \quad 2 \quad \pi \quad N \quad 2 \]

\[ W_4^1 = e^{-j2\pi(1)/2} = e^{-j\pi} = \cos(\pi/2) - j \sin(\pi/2) = -j \]

\[ j \quad 2 \quad \pi \quad 2 \]

\[ W_8^2 = e^{-j2\pi(2)/2} = e^{-j\pi} = \cos(\pi/2) - j \sin(\pi/2) = -j \]
Calculation of Twiddle factor values

\[ W_8^1 = e^{-j \frac{2\pi}{8}(1)} = e^{\frac{j\pi}{4}} = \cos\frac{\pi}{4} - j\sin\frac{\pi}{4} = 0.707 - j0.707 \]

\[ W_8^3 = e^{-j \frac{2\pi}{8}(3)} = e^{\frac{-j3\pi}{4}} = \cos\frac{3\pi}{4} - j\sin\frac{3\pi}{4} = -0.707 - j0.707 \]
Radix-2 DIF FFT algorithm

• In DIF algorithm, the frequency domain sequence $X(k)$ is decimated.
• In this algorithm, N-point time domain sequence is converted to two number of N/2 point sequence.
• Then each N/2 point sequence is converted into two number of N/4 point sequence.
• This process is continued until N/2 number of 2 point sequences is obtained.
Radix-2 DIF FFT algorithm  Cont..

- Normal order
  - $x(0)$-000
  - $x(1)$-001
  - $x(2)$-010
  - $x(3)$-011
  - $x(4)$-100
  - $x(5)$-101
  - $x(6)$-110
  - $x(7)$-111

- Bit reverse order
  - $x(0)$-000
  - $x(4)$-100
  - $x(2)$-010
  - $x(6)$-110
  - $x(1)$-001
  - $x(5)$-101
  - $x(3)$-011
  - $x(7)$-111

Input should be in normal order.
Radix-2 DIF FFT algorithm

\[ X(k) \text{ N point} \]

\[ (N/2) \text{ pt } C_{11}(K) = X(2k) \]

\[ D_{11}(k) \quad D_{12}(k) \]

\[ = C_{11}(2k) = C_{11}(2kH) \]

\[ X(2K + 1) = C_{12}(k) \]

\[ D_{21}(k) \quad D_{22}(k) \]

\[ = C_{12}(2k) = C_{12}(2k+1) \]
Butterfly diagram for 8-point radix-2 DIF FFT algorithm

x(0) → X(0) with $w_2^0$

x(1) → X(4) with $w_4^0$, $-1$

x(2) → X(2) with $w_4^1$

x(3) → X(6) with $w_8^0$, $-1$

x(4) → X(1) with $w_8^1$, $w_4^1$, $-1$

x(5) → X(5) with $w_8^2$

x(6) → X(3) with $w_8^3$, $w_4^1$, $w_2^0$

x(7) → X(7) with $w_8^4$, $-1$
Radix-2 DIF FFT algorithm
Butterfly diagram for 4-point radix-2 DIF FFT algorithm
## Differences between DIT & DIF FFT algorithm

<table>
<thead>
<tr>
<th>DIT FFT</th>
<th>DIF FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time domain sequence is decimated</strong></td>
<td><strong>Frequency domain sequence is decimated</strong></td>
</tr>
<tr>
<td><strong>Input should be in bit reversed order &amp; output will be in normal order</strong></td>
<td><strong>Input should be in normal order &amp; output will be in bit reversed order</strong></td>
</tr>
<tr>
<td><strong>Complex multiplication takes place before the add-subtract operation</strong></td>
<td><strong>Complex multiplication takes place after the add-subtract operation</strong></td>
</tr>
</tbody>
</table>
Similarities in DIT & DIF algorithm

• For both algorithms the value of N should be such that \( N = 2^M \) & there will be M stages of butterfly computation with \( N/2 \) butterfly per stage.

• Both algorithms involve same number of operations. The total no. of complex additions are \( N \log_2 N \) & the total no. of complex multiplications are \( \frac{N}{2} \log_2 N \).

• Both algorithm require bit reversal at some place during computation.
### Relationship between exponential forms and twiddle factors (W) for Periodicity = N

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Exponential form</th>
<th>Symbolic form</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>$e^{-j2\pi n/N} = e^{-j2\pi (n+N)/N}$</td>
<td>$W_{Nn} = W_{Nn+N}$</td>
</tr>
<tr>
<td>02</td>
<td>$e^{-j2\pi (n+N/2)/N} = -e^{-j2\pi n/N}$</td>
<td>$W_{Nn+N/2} = -W_{Nn}$</td>
</tr>
<tr>
<td>03</td>
<td>$e^{-j2\pi k} = e^{-j2\pi Nk/N} = 1$</td>
<td>$W_{NN+K} = 1$</td>
</tr>
<tr>
<td>04</td>
<td>$e^{-j2(2\pi/N)} = e^{-j2\pi/(N/2)}$</td>
<td>$W_{N2} = W_{N/2}$</td>
</tr>
</tbody>
</table>
Matrix Relations

- The DFT samples defined by

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

can be expressed in \( N \times N \) matrix as

\[
\begin{bmatrix}
X(k)
\end{bmatrix} = \left[ \sum_{n=0}^{N-1} W_N^{nk} \right] \begin{bmatrix}
x(n)
\end{bmatrix}
\]

where

\[
\begin{bmatrix}
x
\end{bmatrix} = \begin{bmatrix}
x[0] & x[1] & \cdots & x[N-1]
\end{bmatrix}^T
\]

\[
\begin{bmatrix}
x
\end{bmatrix} = \begin{bmatrix}
x[0] & x[1] & \cdots & x[N-1]
\end{bmatrix}^T
\]
Matrix Relations

\[ \sum_{k=0}^{N-1} W^{nk}_{N} \] can be expanded as \( NXN \) DFT matrix

\[
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & W^1_N & W^2_N & W^3_N \\
1 & W^2_N & W^4_N & W^{N-1}_N \\
1 & W^{(N-1)}_N & W^{2(N-1)}_N & W^{(N-1)^2}_N \\
\end{bmatrix}
\]
• Likewise, the IDFT is

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

can be expressed in NxN matrix form as

\[ x[n] = \left[ \sum_{n=0}^{N-1} W_N^{nk} \right]^{-1} \begin{bmatrix} X(k) \end{bmatrix} \]
Matrix Relations

\[ \sum_{k=0}^{N-1} W^{-nk}_N \text{can also be expanded as NXN DFT matrix} \]

\[ \sum_{k=0}^{N-1} W^{-nk}_N = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W^{-1}_N & W^{-2}_N & \cdots & W^{-(N-1)}_N \\ 1 & W^{-2}_N & W^{-4}_N & \cdots & W^{-2(N-1)}_N \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \sum_{k=0}^{N-1} W^{-nk}_N & \sum_{k=0}^{N-1} W^{-2nk}_N & \cdots & \sum_{k=0}^{N-1} W^{-(N-1)^2}_N \end{bmatrix} \]

Observe:

\[ \sum_{k=0}^{N-1} W^{-nk}_N = \left[ \frac{1}{N} \sum_{n=0}^{N-1} W^{nk}_N \right]^* \]

The inversion can be had by Hermitian conjugating \( j \) by \( -j \) and dividing by \( N \).
Zero-Padding

• Consider an \( L \)-point input sequence \( x(n) \) and a \( P \)-point impulse response \( h(n) \).

• The linear convolution of these two sequences \( y(n) \) has finite duration with length \( (L+P-1) \).

• For the circular convolution and linear convolution to be identical, the circular convolution must have a length of at least \( (L+P-1) \) points.
Zero-Padding

• The circular convolution can be achieved by multiplying the DFTs of \( x(n) \) and \( h(n) \).

• Since the length of the linear convolution is \( (L+P-1) \) points, the DFTs that we compute must also be of at least that length, i.e., both \( x(n) \) and \( h(n) \) must augmented with sequence values of zero.

• The process is called **Zero-Padding**
Zero padding

• While finding N-point DFT of x(n), if the length of x(n) is M then, we should add N-M number of zeros in x(n).
• This is called zero padding
• Uses:
  – Frequency spectrum is good
  – DFT is used in linear filtering because of zero padding