Rajshahi University of Engineering & Technology

Department : Electrical & Computer Engineering

Course No: ECE 4124

Course Name: Digital Signal Processing Sessional

Submitted By:

Sumaiya Tabassum

Roll: 1810023

Submitted To:

Md. Nahiduzzaman

Lecturer,Dept of ECE

Experiment No:7

Experiment Date:9.7.23

Experiment Name: Computation of DFT using DIT-FFT algorithm.

Theory: FFT is a very efficient algorithm in computing DFT coefficients and can reduce a very large amount of computational complexity (multiplications).FFT algorithm is divided into two part i.e. decimation in time (DIT) and decimation in frequency (DIF). In DIT algorithm firstly computed multiplier then adder but in DIF firstly computed adder then multiplier. In decimation-in-time technique ,we split the input into odd and even parts. In the context of fast Fourier transform algorithms, a butterfly is a portion of the computation that combines the results of smaller discrete Fourier transforms (DFTs) into a larger DFT, or vice versa. Though it is not the efficient algorithm, it lays foundation for time-efficient DFT calculations. the conventional DFT algorithm has a high computational complexity, making it inefficient for practical applications.To overcome this limitation, the Fast Fourier Transform (FFT) algorithm was developed. The FFT algorithm efficiently computes the DFT by exploiting the inherent symmetry and repetitive nature of the complex exponential functions involved. By using clever algorithms and techniques, the FFT reduces the computational complexity from O(N^2) to O(N log N).

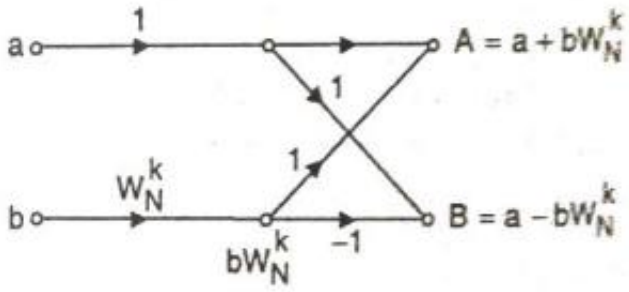

Fig:basic butterfly computation

Procedure (by DIT-FFT method):

Steps:

1 . First, we determine whether the length of the given array is even or not. If not, then we make it even by adding zeros behind it. To do this we first compute whether the size of the array is nearest to any power of 2. We then proceed to add zeros to the array accordingly.

2.Then we compute the array size and number of conversion stages. The no. of conversion stages is given by log of N to the base 2 where N is the size of the array. Followed by the bit reversal of the input sequence. The number of points N=2^m, where the stages m=log2 N

3 .We initialise the start points of the butterfly structure. We introduce a variable n which would determine the number of butterfly structures that are occurring in the algorithm.

4. We introduce the twiddle factor. We multiply and add the upper points of the butterfly structure, subsequently we multiply and subtract from the lower points of the butterfly structure. Then we increment the points as well as n for the next structure.

5 .We introduce a condition that would discontinue the butterfly structure and move on to the next stage.

Code:

```
input_sequence = input('input: '); % Example input sequence
output_sequence = compute(input_sequence); % Compute the FFT using the
butterfly DIT algorithm
disp(output_sequence); % Display the computed FFT

 %compute magnitude and phase
mag_X=zeros(1,length(output_sequence));
phi_X=zeros(1,length(output_sequence));
for k=1:length(output_sequence)
    mag_X(k)=sqrt((output_sequence(k))^2+imag(output_sequence(k))^2);
    phi_X(k)=atan2(imag(output_sequence(k)),real(output_sequence(k)));
end
%plot results

subplot(2,1,1);
stem(mag_X);
xlabel('Frequency');
ylabel('Magnitude');
title('Magnitude spectrum');
subplot(2,1,2);
stem(phi_X);
xlabel('Frequency(Hz)');
ylabel('phase(rad)');
title('Phase Spectrum');

 function a=compute(x)
p=nextpow2(length(x)); %determining the length closest to the power of
2
x=[x zeros(1,((2^p)-length(x)))]; %padding with zeros if necessary
N=length(x); %Length of the input array
S=log2(N); %determining the no. of stages
x=bitrevorder(x); %reversing the bit order of the input sequence
for stage=1:S %iterating through each stage
 p=1; %initialising the butterfly sequence
 q=1+2^(stage-1);
 n=0;
while(n<=(2^(stage-1)-1) && q<=N)
 w=exp((-1i)*2*pi*n/(2^stage)); %Incorporating the twiddle factor
 y=x(p)+w*x(q); %Equation for 1st part of butterfly operation
 z=x(p)-w*x(q); %Equation for 2nd part of butterfly operation
 x(p)=y; %Equating
 x(q)=z;
 p=p+1; % Incrementing the sequences for next butterfly structure
```

```
 q=q+1;
 n=n+1;
if(rem(q,2^stage)==1) % repetition of the butterfly
 p=p+2^(stage-1);% structure for other input sequences
 q=q+2^(stage-1);
 n=0;
end
end
end
a=x;
 end
```

Output:

```
input: [1 2 3 4 5 6 7 8]
  Columns 1 through 4

  36.0000 + 0.0000i  -4.0000 + 9.6569i  -4.0000 + 4.0000i  -4.0000 + 1.6569i

  Columns 5 through 8

  -4.0000 + 0.0000i  -4.0000 - 1.6569i  -4.0000 - 4.0000i  -4.0000 - 9.6569i
```

Plot Result:

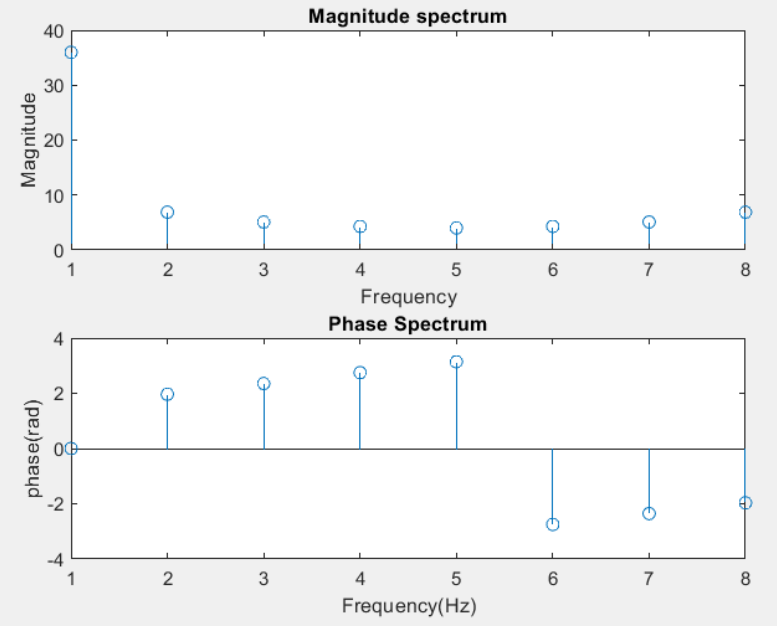

Fig:Magnitude and Phase

Conclusion:

The FFT function returns the spectrum so that the DC (constant/average) value is first in the array.This experiment provided us with practical insights into the theory and implementation of the FFT algorithm.Understanding the FFT's capabilities and limitations equips us with the necessary knowledge to apply it effectively in various signal processing tasks. By implementing the FFT algorithm in our experiment, we were able to efficiently compute the DFT of discrete time-domain signals. This significantly reduced the computational complexity compared to the traditional DFT approach. The speed and efficiency of the FFT make it an essential tool for various applications, such as signal processing, audio and image compression, and spectral analysis.

Reference:

https://uomustansiriyah.edu.iq/media/lectures/5/5_2021_12_15!05_17_30_PM.pdf

https://www.rcet.org.in/uploads/academics/rohini_39437371635.pdf