# Digital Signal Processing [Lab-8]

## Table of Contents

- Authors: *Kshitij Srivastava(1510110200)* and *Nilambar Saha(1510110246)*
- Lab Instructor: *Dr. Ravi Kant Saini*
- Batch: ECE32 Friday 1pm to 3pm
- Email: ks435@snu.edu.in

# Objective:

```
%DIT-FFT Algorithm (In this Algorithm, we use DIT-FFT and DIF-FFT Algorithm
%to compute DFT with very less time compelexity. This was then used to see
%the power spectral density)
```

# Program:

```
clc;
clear all;
close all;

% * |*Matlab Commands for finding the DFT by DIT-FFT Algorithm*|

x=[2,0,2,0,2,0,2,0];
length=8;%Length of the input
levels=log2(8);%no of multiplication and addition required

index=0:7; %index value of variable x
bit_index=zeros(1,8);%Initializing the variable bit_index
bit_index=bitrevorder(index); % To store the bit reversed index of x

for i=1:8
    x_new(i)=x(bit_index(i)+1); %Finding the array from bit reversed indexes
end

input=x_new;%input to the level is the array from bit reversed indexes
for i=1:3 %Number of levels in DIT

    %W_upper for array with value 1 to be multiplied to upper blocks
    W_upper=ones( 1, 2^(i-1) ); % 2^(level-1) where level=i

    for j=0:( (2^i) /2)-1  %For finding the index for W_lower (j)
```

```
        W_found=W( 2^i,j);% W(2^i,2^level-number)
        W_lower(j+1)=W_found;%W_lower stored
    end
    Wxxx=horzcat(W_upper,W_lower);%Concatenating the W_upper and W_lower
    %to create a variable Wxxx to be multiplied to a block

    %To multiply W with the blocks
    index_plus=(2^i)-1;% Variable to find till what index the block
    %has to be taken
    for k=1:(2^i):8 % Varible k for finding the starting of the block at
        %a level i
        sub_block=input(k:k+index_plus ); %Finding the block from the input
        x_temp(k:k+index_plus)=sub_block.*Wxxx; % Multiplying W with the block
    end

    %Butterfly Addition and Subtraction
    new_block=[];%Creating a empty array for output after every level
    for j=1:(2^i):8 %For finding the starting vertices of every block
        block_index=(2^i)-1;  %(2^i)-1, For finding the end of each block
        block=x_temp(j:j+block_index);%Selecting a block from above step

        add_index=(2^(i-1) )-1; %(2^(level-1) )-1  level=i
        add_block=block(1:1+add_index);%Finding the addition block from the
        %block where addition will take place
        diff_block=block(1+add_index+1:2+2*add_index);%Finding the
        %difference block from block where subtraction will take place

        new_add_block=add_block+diff_block; %Additions
        new_diff_block=add_block-diff_block; %Subtractions
        new_block=horzcat(new_block,new_add_block,new_diff_block);
        %Horizontally concatenating the new_block, new_add_block and
        %new_diff_block to create output of each level
    end
     input=new_block;%Output of each level becomes input to next level
end

xfft_dit=round(new_block);%Rounding the output to get final output
```

- **Matlab Commands for finding the DFT by DIF-FFT Algorithm**

```
clc;
clear all;
close all;

x=[2,0,2,0,2,0,2,0];
length=8;%Length of the input
levels=log2(8);%no of multiplication and addition required

input=x; %input to the first level
for i=1:3 %Number of levels in DIF

    %W_upper for array with value 1 to be multiplied to upper blocks
    W_upper=ones( 1, 2^(i-1) ); % 2^(level-1) where level=i
    for j=0:( (2^i) /2)-1  %For finding the index for W_lower (j)
```

```
        W_found=W( 2^i,j);% W(2^i,2^level-number)
        W_lower(j+1)=W_found;%W_lower stored
    end
    Wxxx=horzcat(W_upper,W_lower); %Concatenating the W_upper and W_lower
    %to create a variable Wxxx to be multiplied to a block

    %To multiply W with the blocks
    index_plus=(2^i)-1; % Variable to find till what index the block
    %has to be taken
    for k=1:(2^i):8  %Varible k for finding the starting of the block at
        % level i
        sub_block=input(k:k+index_plus ); %Finding the block from the input
        x_temp(k:k+index_plus)=sub_block.*Wxxx; % Multiplying W with the block
    end

    %Butterfly Addition and Subtraction
    new_block=[]; %Creating a empty array for output after every level
    for j=1:(2^i):8 %For finding the starting vertices of every block
        block_index=(2^i)-1;  %(2^i)-1, For finding the end of each block
        block=x_temp(j:j+block_index); %Selecting a block from above step

        add_index=(2^(i-1) )-1; %(2^(level-1) )-1  level=i
        add_block=block(1:1+add_index); %Finding the addition block from the
        %block where addition will take place
        diff_block=block(1+add_index+1:2+2*add_index); %Finding the
        %difference block from block where subtraction will take place

        new_add_block=add_block+diff_block; %Additions
        new_diff_block=add_block-diff_block; %Subtractions
        new_block=horzcat(new_block,new_add_block,new_diff_block);
        %Horizontally concatenating the new_block, new_add_block and
        %new_diff_block to create output of each level
    end
     input=new_block; %Output of each level becomes input to next level
end

%Bit reversal
index=0:7; %index value after butterfly addition and subtractions
bit_index=zeros(1,8); %Initializing the variable bit_index
bit_index=bitrevorder(index); % To store the bit reversed index of index

%
  for i=1:8
      x_new(i)=new_block(bit_index(i)+1);
      %Finding the array from bit reverse index of array got from butterfly
      %addition and subtraction
  end

xfft_dif=round(x_new);%Rounding of the output fft to find the output fft
```

# Results for Q1:

- **Plot for the Question No 1**

```
xfft_dit

        Undefined function or variable 'xfft_dit'.

        Error in Lab8_new (line 142)
        xfft_dit
```

- **Matlab Commands for finding the DFT by DIF-FFT Algorithm**

```
clc;
clear all;
close all;

x=[2,0,2,0,2,0,2,0];
length=8;%Length of the input
levels=log2(8);%no of multiplication and addition required

input=x; %input to the first level
for i=1:3 %Number of levels in DIF

    %W_upper for array with value 1 to be multiplied to upper blocks
    W_upper=ones( 1, 2^(i-1) ); % 2^(level-1) where level=i
    for j=0:( (2^i) /2)-1  %For finding the index for W_lower (j)
        W_found=W( 2^i,j);% W(2^i,2^level-number)
        W_lower(j+1)=W_found;%W_lower stored
    end
    Wxxx=horzcat(W_upper,W_lower); %Concatenating the W_upper and W_lower
    %to create a variable Wxxx to be multiplied to a block

    %To multiply W with the blocks
    index_plus=(2^i)-1; % Variable to find till what index the block
    %has to be taken
    for k=1:(2^i):8  %Varible k for finding the starting of the block at
        % level i
        sub_block=input(k:k+index_plus ); %Finding the block from the input
        x_temp(k:k+index_plus)=sub_block.*Wxxx; % Multiplying W with the block
    end

    %Butterfly Addition and Subtraction
    new_block=[]; %Creating a empty array for output after every level
    for j=1:(2^i):8 %For finding the starting vertices of every block
        block_index=(2^i)-1;  %(2^i)-1, For finding the end of each block
        block=x_temp(j:j+block_index); %Selecting a block from above step

        add_index=(2^(i-1) )-1; %(2^(level-1) )-1  level=i
        add_block=block(1:1+add_index); %Finding the addition block from the
        %block where addition will take place
        diff_block=block(1+add_index+1:2+2*add_index); %Finding the
        %difference block from block where subtraction will take place

        new_add_block=add_block+diff_block; %Additions
        new_diff_block=add_block-diff_block; %Subtractions
        new_block=horzcat(new_block,new_add_block,new_diff_block);
        %Horizontally concatenating the new_block, new_add_block and
        %new_diff_block to create output of each level
    end
     input=new_block; %Output of each level becomes input to next level
end

%Bit reversal
index=0:7; %index value after butterfly addition and subtractions
bit_index=zeros(1,8); %Initializing the variable bit_index
bit_index=bitrevorder(index); % To store the bit reversed index of index

%
  for i=1:8
      x_new(i)=new_block(bit_index(i)+1);
      %Finding the array from bit reverse index of array got from butterfly
      %addition and subtraction
  end

xfft_dif=round(x_new);%Rounding of the output fft to find the output fft
```

# Results for Q2:

- **Plot for the Question No 2**

```
xfft_dif
```

- **Matlab Commands for finding the DFT of ECG signal by DIT_FFT**

```
clc;
clear all;
close all;

load('inputData.mat');
inputData=transpose(inputData);
x=inputData;

length=1024;%Length of the input
levels=log2(1024);%no of multiplication and addition required

index=0:1023; %index value of variable x
bit_index=zeros(1,1024);%Initializing the variable bit_index
bit_index=bitrevorder(index); % To store the bit reversed index of x

for i=1:1024
    x_new(i)=x(bit_index(i)+1); %Finding the array from bit reversed indexes
end

input=x_new;%input to the level is the array from bit reversed indexes
for i=1:levels %Number of levels in DIT

    %W_upper for array with value 1 to be multiplied to upper blocks
    W_upper=ones( 1, 2^(i-1) ); % 2^(level-1) where level=i

    for j=0:( (2^i) /2)-1  %For finding the index for W_lower (j)
        W_found=W( 2^i,j);% W(2^i,2^level-number)
        W_lower(j+1)=W_found;%W_lower stored
    end
    Wxxx=horzcat(W_upper,W_lower);%Concatenating the W_upper and W_lower
    %to create a variable Wxxx to be multiplied to a block

    %To multiply W with the blocks
    index_plus=(2^i)-1;% Variable to find till what index the block
    %has to be taken
    for k=1:(2^i):1024 % Varible k for finding the starting of the block at
        %a level i
        sub_block=input(k:k+index_plus ); %Finding the block from the input
        x_temp(k:k+index_plus)=sub_block.*Wxxx; % Multiplying W with the block
    end

    %Butterfly Addition and Subtraction
    new_block=[];%Creating a empty array for output after every level
    for j=1:(2^i):1024 %For finding the starting vertices of every block
        block_index=(2^i)-1;  %(2^i)-1, For finding the end of each block
        block=x_temp(j:j+block_index);%Selecting a block from above step

        add_index=(2^(i-1) )-1; %(2^(level-1) )-1  level=i
        add_block=block(1:1+add_index);%Finding the addition block from the
        %block where addition will take place
        diff_block=block(1+add_index+1:2+2*add_index);%Finding the
        %difference block from block where subtraction will take place

        new_add_block=add_block+diff_block; %Additions
        new_diff_block=add_block-diff_block; %Subtractions
        new_block=horzcat(new_block,new_add_block,new_diff_block);
        %Horizontally concatenating the new_block, new_add_block and
        %new_diff_block to create output of each level
    end
     input=new_block;%Output of each level becomes input to next level
end

xfft=round(new_block);%Rounding the output to get final output
```

- **Matlab Commands for normalised magitude square of DFT Coefficients**

```
xfft_mag_square=abs(xfft).^2;%Finding the square of absolute fft
PSD_max=max(xfft_mag_square);%Finding the maximum term from the PSD
PSD_max_arr=zeros(1,1024);

for k=1:1024
    PSD_max_arr(k)=PSD_max;%Forming an array with same value of PSD_max
end

norm_PSD=xfft_mag_square./PSD_max_arr; %Element-wise division to get
%normalised PSD
```

- *Matlab Commands to calculate 3db power bandwidth *

```
thousand_normPSD=round(1000*norm_PSD);% Nomrmalised PSD
three_db_value=700; %three db value =1000*0.7
threedb_count=0;
```

```
for i=1:1024
    if thousand_normPSD(i)>700 && thousand_normPSD(i)<800
        threedb_count=threedb_count+1;
        three_db_index(threedb_count)=i;
    end
end
low_freq_db=(three_db_index(1)/1024)*500;
high_freq_db=(three_db_index(2)/1024)*500;
```

- *Matlab Commands for frequency range for 90% of normalized power. *

```
sum_norm_PSD=sum(norm_PSD); %Summing of the normailsed PSD
sum_norm_PSD=round(sum_norm_PSD); %Rounding of the sum of normaalised PSD
limit=0.9*sum_norm_PSD; % Finding the limit for the 90% of the
%normalised power
limit=round(limit);%Rounding off that limit

count=0;
for i=1:1024 %for the left_most index
    for j=1:1024 %For the high_most index (high frequency)
        sums=round(sum( norm_PSD(i:j) ));%Summing norm PSD from i to j
        if sums==limit %If the summing of norm PSD from i to j==limit
           count=count+1; %Counter to count when sum becomes equal to limit
           left_index(count)=i; %To store the left index in the array
           right_index(count)=j; %To store the right most index in the array
        end
    end
end

low_freq=(1*500)/1024;%Finding the low frequency
high_freq=(1021*500)/1024;%Finding the high frequency range
```

# Results for Q3:

- **Plot for the Question No 3 a**

```
figure;plot(norm_PSD);
title('PSD of ECG Signal');xlabel('Index');ylabel('Amplitude');
```

- **Result for the Question No 3 b**

```
%low index=
 three_db_index(1)
%high index=
 three_db_index(2)
%low freq , high-freq=
 low_freq_db, high_freq_db
%bandwidth
high_freq_db-low_freq_db
```

- **Result for the Question No 3 c**

```
%low freq , high-freq=
 low_freq, high_freq
```

```
%bandwidth
high_freq-low_freq
```

*Published with MATLAB® R2014a*