INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, NAGPUR

HARDWARE DESCRIPTIVE LANGUAGE PROJECT REPORT

Implementaion of IDFT using DIT FFT and DIF FFT algorithms

Submitted To:
Dr. Vipin Kamble

Submitted By:
Kautilya Joshi (BT17ECE009)
Utkarsh Bhiogade (BT17ECE035)
Kritika Dhawale (BT17ECE042)

April 1st, 2019

ACKNOWLEDGEMENT

We express our deep sense of gratitude to our guide professor, $\boldsymbol{Dr.VipinKamble}$, for his valuable guidance and encouragement. We would also like to express our gratitude to all those who were involved directly or indirectly with the completion of this project.

Contents

1 Abstract **4**

2 Introduction **4**

3 The Approach towards 8-point IDFT **4**

3.1 FFT Algorithm 4

3.2 Radix-2 Algorithm 4

3.3 Derivation of IDFT 5

3.3.1 Derivation of DIT IDFT 5

3.3.2 Derivation of DIF IDFT 7

4 Code for IDFT of DIT and DIF using FFT **9**

4.1 IDFT Using DIT FFT 9

4.1.1 Package 9

4.1.2 Main Code 10

4.1.3 Butterfly Structure 11

4.1.4 Testbench 12

4.2 IDFT Using DIF FFT 13

4.2.1 Package 13

4.2.2 Main Code 14

4.2.3 Butterfly Structure 15

4.2.4 Testbench 16

5 Code Explanation **17**

5.1 Package 17

5.2 Main code 17

6 Observation **18**

7 Conclusion **18**

8 References **18**

1 Abstract

Discrete Fourier Transform (DFT) and Inverse Fourier Transform (IDFT) are a fundamental digital signal processing algorithm are used in many applications, including frequency analysis and frequency domain processing. In this project we have implemented an 8-point IDFT using Decimation in Time(DIT) FFT and Decimation in Frequency(DIF) FFT.

2 Introduction

Fast Fourier Transform (FFT) and its inverse (IFFT) play a significant role in many digital signal processing applications. It is the decomposition of a sampled signal in terms of sinusoidal (complex exponential) components. The symmetry and periodicity properties of the DFT are exploited to significantly lower its computational requirements. The resulting algorithms are known as Fast Fourier Transforms (FFTs). It has been applied in a large range of fields and applications such as Asymmetrical Digital Subscriber Line (ADSL), Digital Audio Broadcasting (DAB), Digital Video Broadcasting (DVB) and Orthogonal Frequency Division Multiplexing (OFDM) systems.

3 The Approach towards 8-point IDFT

3.1 FFT Algorithm

The FFT/IFFT is derived from the main function DFT (Discrete Fourier Transform). The computation for N-points of the DFT will be calculated one by one for each point while for FFT/IFFT, the computation is done simultaneously which saves quite a lot of time.

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)W_N^{-kn} \qquad k = 1, 2, 3, ...N-1$$

3.2 Radix-2 Algorithm

Useful when N is a power of 2: $N = r^v$for integers r and v. r is called the radix, which comes from the Latin word meaning "a root," and has the same origins as the word radish. When N is a power of r = 2, this is called radix-2, and the natural "divide and conquer approach" is to split the sequence into two sequences of length N/2.

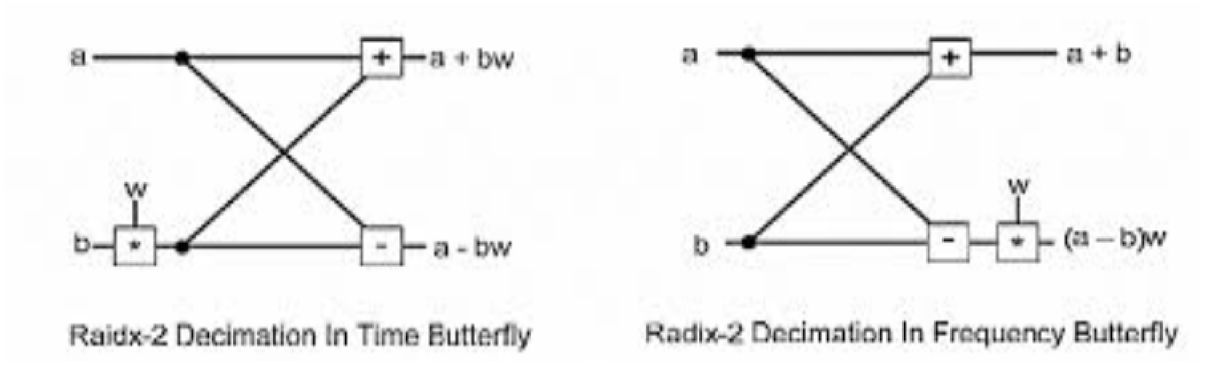

a
a + bw
w
b
a - bw
Raidx-2 Decimation In Time Butterfly
a
a + b
w
b
(a – b)w
Radix-2 Decimation In Frequency Butterfly

Figure 1: Radix-2

3.3 Derivation of IDFT

3.3.1 Derivation of DIT IDFT

Proof. Let us consider the computation of the $N = 2^v$ point IDFT by the divide-and conquer approach. We split the N-point data sequence into two N/2-point data sequences $f_1(n)$ and $f_2(n)$, corresponding to the even-numbered and odd-numbered samples of $x(n)$, respectively, that is,

$$f_1(n) = x(2n)$$
$$f_2(n) = x(2n+1) \qquad n = 1, 2, 3, ...\frac{N}{2} - 1$$

Thus f1(n) and f2(n) are obtained by decimating x(n) by a factor of 2, and hence the resulting FFT algorithm is called a decimation-in-time algorithm. Now the N-point IDFT can be expressed in terms of the IDFT's of the decimated sequences as follows:

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)W_N^{-kn} \qquad k = 1, 2, 3, ...N-1$$
$$X(k) = \frac{1}{N}\sum_{n=even} x(n)W_N^{-kn} + \frac{1}{N}\sum_{n=odd} x(n)W_N^{-kn}$$
$$X(k) = \frac{1}{N}\sum_{m=0}^{N/2-1} x(2m)W_N^{-k2m} + \frac{1}{N}\sum_{m=0}^{N/2-1} x(2m+1)W_N^{-k(2m+1)}$$

But $W_N^2 = W_{N/2}$. With this substitution, the equation can be expressed as

$$X(k) = \frac{1}{N}[\sum_{m=0}^{N/2-1} f_1(m)W_{N/2}^{-km} + W_{N/2}^{-k}\sum_{m=0}^{N/2-1} f_2(m)W_{N/2}^{-km}]$$
$$X(k) = \frac{1}{N}[F_1(k) + W_{N/2}^{-k}F_2(k)], \qquad k = 0, 1, 2, ...N-1$$

where $F_1(k)$ and $F_2(k)$ are the N/2-point IDFTs of the sequences $f_1(m)$ and $f_2(m)$, respectively.

Since $F_1(k)$ and $F_2(k)$ are periodic, with period N/2, we have $F_1(k+N/2) = F_1(k)$and $F_2(k+N/2) = F_2(k)$. In addition, the factor $W_N^{-k+N/2} = -W_N^{-k}$. Hence the equation may be expressed as

$$X(k) = \frac{1}{N}[F_1(k) + W_{N/2}^{-k} F_2(k)], \qquad k = 0, 1, 2, ...N-1$$

$$X(k+\frac{N}{2}) = \frac{1}{N}[F_1(k) - W_{N/2}^{-k} F_2(k)], \qquad k = 0, 1, 2, ...N-1$$

$\square$

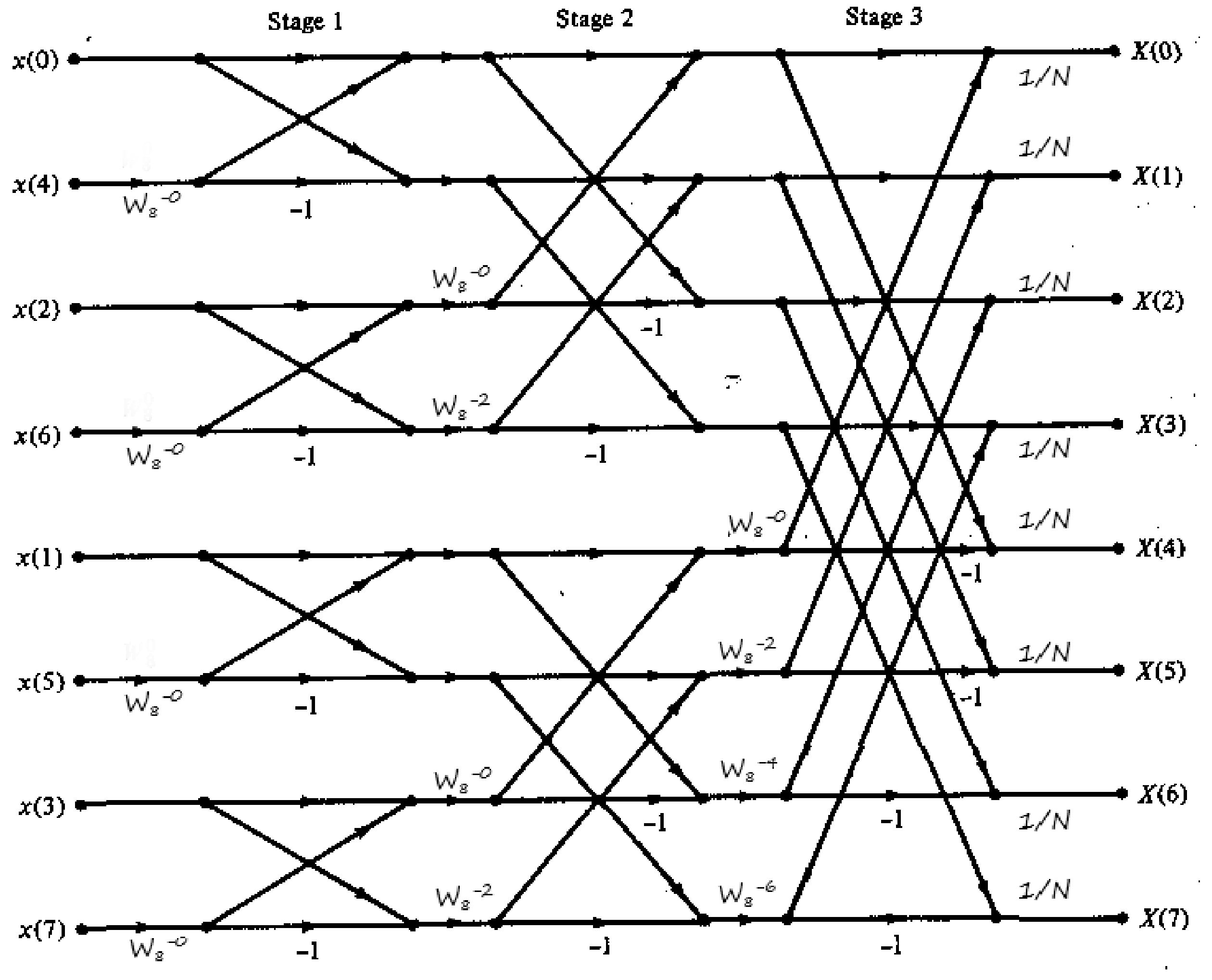


Figure 2: 8 point DIT IDFT Butterfly Structure

3.3.2 Derivation of DIF IDFT

Proof. To derive the algorithm, we begin by splitting the IDFT formula into two summations, one of which involves the sum over the first N/2 data points and the second sum involves the last N/2 data points. Thus we obtain

$$X(k) = \frac{1}{N}[\sum_{n=0}^{(N/2)-1} x(n)W_N^{-kn} + \sum_{n=N/2}^{N-1} x(n)W_N^{-kn}]$$

$$X(k) = \frac{1}{N}[\sum_{n=0}^{N/2-1} x(n)W_N^{-kn} + W_N^{-\frac{kN}{2}} \sum_{n=0}^{N/2-1} x(n+N/2)W_N^{-kn}]$$

$$\textit{Since} \quad W_N^{-\frac{kN}{2}} = (-1)^k$$

$$X(k) = \frac{1}{N}\sum_{n=0}^{N/2-1} [x(n) + (-1)^k x(n+\frac{N}{2})]W_N^{-kn}]$$

Now, let us split (decimate) X(k) into the even- and odd-numbered samples. Thus we obtain

$$X(2k) = \frac{1}{N}\sum_{n=0}^{N/2-1} [x(n) + x(n+\frac{N}{2})]W_{\frac{N}{2}}^{-kn}] \qquad k = 0, 1, ..., \frac{N}{2} - 1$$

$$X(2k+1) = \frac{1}{N}\sum_{n=0}^{N/2-1} [x(n) - x(n+\frac{N}{2})]W_{\frac{N}{2}}^{-kn}W_N^{-n}] \qquad k = 0, 1, ..., \frac{N}{2} - 1$$

where we have used the fact that $W_N^2 = W_{N/2}$. □

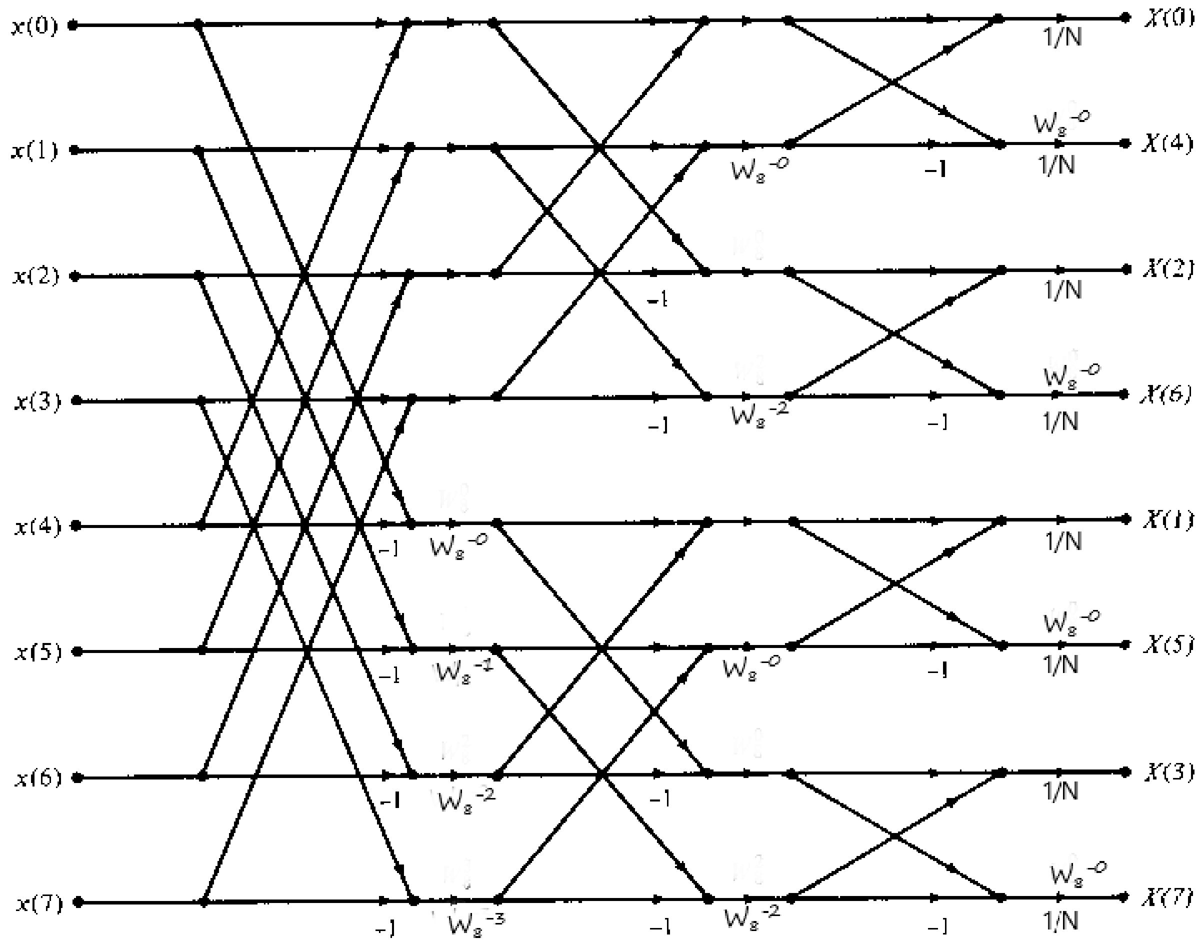


Figure 3: 8 point DIF IDFT Butterfly Structure

4 Code for IDFT of DIT and DIF using FFT

4.1 IDFT Using DIT FFT

4.1.1 Package

```vhdl
library IEEE;                                   --IMPORTING LIBRARY
use IEEE.std_logic_1164.all;
use IEEE.MATH_REAL.ALL;                         --USING MATH_REAL LIBRARY
----------------------------------------------------------------------------
package dit_ifft_pkg is                         --PACKAGE DECLARATION
type complex is                                 --DEFINING A DATA STRUCTURE
    record                                      --DEFINING RECORD
        r : real;
        i : real;
    end record;

--DECALRING AN ARRAY OF TYPE COMPLEX OF LENGTH = 8
type ar is array (0 to 7) of complex;
--DECLARING AN ARRAY OF TYPE COMPLEX OF LENGTH = 4
type ar2 is array (0 to 3) of complex;

--FUNCTION DECLARATION OF ADDITION
function add (n1,n2 : complex) return complex;
--FUNCTION DECLARATION OF SUBSTRACTION
function sub (n1,n2 : complex) return complex;
--FUNCTION DECLARATION OF MULTIPLICATION
function multi (n1,n2 : complex) return complex;

end dit_ifft_pkg;
----------------------------------------------------------------------------
package body dit_ifft_pkg is                    --START OF PACKAGE BODY
----------------------------------------------------------------------------
--FUNCTION FOR ADDITION
function add (n1,n2 : complex) return complex is
variable s : complex;                           --VARIABLE DECLARATION
begin
s.r:=n1.r + n2.r;                             --ADDITION OF REAL PARTS
s.i:=n1.i + n2.i;                             --ADDITOIN OF IMAGINARY PARTS
return s;                                     --RETURNING SUM
end add;
----------------------------------------------------------------------------
----------------------------------------------------------------------------
--FUNCTION FOR SUBSTRACTION
function sub (n1,n2 : complex) return complex is
variable d : complex;                        --VARIABLE DECLARATION
begin
d.r:=n1.r - n2.r;                            --SUBSTRACTING REAL PARTS
d.i:=n1.i - n2.i;                            --SUBSTRACTING IMAGINARY PARTS
return d;                                    --RETURNING SUBSTRACTED VALUE
end sub;
----------------------------------------------------------------------------
```

```
--FUNCTION FOR MULTIPLICATION.
function multi (n1,n2 : complex) return complex is
variable p : complex;                         --VARIABLE DECLARATION
begin
p.r:=(n1.r * n2.r) - (n1.i * n2.i);           --MULTIPLYING REAL WITH ...
    IMAGINARY VALUE
p.i:=(n1.r * n2.i) + (n1.i * n2.r);           --MULTIPLYING IMAGINARY PARTS
return p;                                     --RETURNING MULTIPLIED VALUE
end multi;
--------------------------------------------------------------------------
end dit_ifft_pkg;                             --END OF PACKAGE BODY
```

4.1.2 Main Code

```
library IEEE;                                 --IMPORTING LIBRARY
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.MATH_REAL.ALL;
library work;                                 --USING FILES FROM WORK ...
    DIRECTORY
use work.dit_ifft_pkg.ALL;                    --USING PACKAGE ...
    DIT_IFFT_PKG FROM WORK DIRECTORY
--------------------------------------------------------------------------
entity dit_ifft_8pt is                        --ENTITY DECLARATION
port(   s : in ar;                            --INPUT SIGNALS
        y : out ar);                          --OUTPUT SIGNALS
end dit_ifft_8pt;
--------------------------------------------------------------------------
architecture Behavioral of dit_ifft_8pt is    --ARCHITECTURE DECLARATION
component butterfly is                        --COMPONENT DECLARATION
   port(
      b1,b2 : in complex;                     --INPUTS OF BUTTERFLY ...
          STRUCTURE
      w :in complex;                          --PHASE FACTOR
      z1,z2 :out complex);                    --OUTPUTS OF BUTTERFLY ...
          STRUCTURE
end component;
-------------------------------------------------------------------------
--DEFINING SIGNLAS Z1 AND Z2 WITH DEFAULT VALUE OF(0.0,0.0)
signal z1,z2 : ar ;

--PHASE FACTOR, w_N = e^(-j*(2*p/N)*n)
--w_N^n = cos((2*p/N)*n) - j*sin((2*p/N)*n);
--w_N VALUES FOR n = 0-4

constant w : ar2 := ( (1.0,0.0), (0.7071,0.7071), (0.0,1.0), ...
    (-0.7071,0.7071) );
signal p:ar;                                 --DEFINING SIGNAL P OF TYPE AR
begin
--FIRST STAGE OF BUTTERFLY, n = 0; N = 8
--MAPPING INPUTS AND OUTPUTS
bfly1 : butterfly port map(s(0),s(4),w(0),z1(0),z1(1));
```

```
bfly2 : butterfly port map(s(2),s(6),w(0),z1(2),z1(3));
bfly3 : butterfly port map(s(1),s(5),w(0),z1(4),z1(5));
bfly4 : butterfly port map(s(3),s(7),w(0),z1(6),z1(7));

--SECOND STAGE OF BUTTERFLY, n = 0,2; N = 8
--MAPPING INPUTS AND OUTPUTS
bfly5 : butterfly port map(z1(0),z1(2),w(0),z2(0),z2(2));
bfly6 : butterfly port map(z1(1),z1(3),w(2),z2(1),z2(3));
bfly7 : butterfly port map(z1(4),z1(6),w(0),z2(4),z2(6));
bfly8 : butterfly port map(z1(5),z1(7),w(2),z2(5),z2(7));

--THIRD STAGE OF BUTTERFLY, n = 0,2,4,6; N = 8
--MAPPING INPUTS AND OUTPUT
bfly9 : butterfly port map(z2(0),z2(4),w(0),p(0),p(4));
bfly10 : butterfly port map(z2(1),z2(5),w(1),p(1),p(5));
bfly11 : butterfly port map(z2(2),z2(6),w(2),p(2),p(6));
bfly12 : butterfly port map(z2(3),z2(7),w(3),p(3),p(7));

y(0)<= multi(p(0),(0.125,0.0));             --DIVIDING BY FACTOR OF 1/N
y(1)<= multi(p(1),(0.125,0.0));             --ASSIGNING TO OUTPUT SIGNAL
y(2)<= multi(p(2),(0.125,0.0));
y(3)<= multi(p(3),(0.125,0.0));
y(4)<= multi(p(4),(0.125,0.0));
y(5)<= multi(p(5),(0.125,0.0));
y(6)<= multi(p(6),(0.125,0.0));
y(7)<= multi(p(7),(0.125,0.0));
end Behavioral;
```

4.1.3 Butterfly Structure

```
library IEEE;                                 --IMPORTING LIBRARY
use IEEE.STD_LOGIC_1164.ALL;
library work;                                 --USING FILES FROM WORK ...
    DIRECTORY
use work.dit_ifft_pkg.ALL;                    --USING PACKAGE ...
    DIT_IFFT_PKG FROM WORK DIRECTORY
-----------------------------------------------------------------------
entity butterfly is                           --ENTITY DECLARATION
   port(
      b1,b2 : in complex;                     --INPUTS OF BUTTERFLY ...
          STRUCTURE
      w :in complex;                          --PHASE FACTOR
      z1,z2 :out complex);                    --OUTPUTS OF LIBRARY
end butterfly;
-----------------------------------------------------------------------
architecture Behavioral of butterfly is       --ARCHITECTURE DECLARATION
begin
z1 <= add(b1,multi(b2,w));              --BUTTERFLY EQUATION FOR ADDITION
z2 <= sub(b1,multi(b2,w));              --BUTTERFLY EQUATION FOR ...
    SUBSTRACTION
end Behavioral;                                   --END OF ARCHITECTURE
```

4.1.4 Testbench

```vhdl
LIBRARY ieee;                               --IMPORTING LIBRARY
USE ieee.std_logic_1164.ALL;
library work;                               --WORKING IN CURRENT(WORK) ... FOLDER
use work.dit_ifft_pkg.all;                  --USING PACKAGE ... DIT_IFFT_PKG FROM WORK DIRECTORY
-----------------------------------------------------------------------------
ENTITY tb IS                                --TESTBENCH ENTITY DECLARATION
END tb;
-----------------------------------------------------------------------------
ARCHITECTURE behavior OF tb IS              --ARCHITECTURE DECLARATION
signal s,y : ar;                            --SIGNAL DECLARATION
begin
-- Instantiating the Unit Under Test (UUT)
uut: entity work.dit_ifft_8pt PORT MAP (s => s,y => y);
   stim_proc: process                       -- Stimulus process
   begin
        s(0) <= (1.0,0.0);                  --GIVING INPUTS
        s(1) <= (2.0,0.0);
        s(2) <= (3.0,0.0);
        s(3) <= (4.0,0.0);
        s(4) <= (5.0,0.0);
        s(5) <= (6.0,0.0);
        s(6) <= (7.0,0.0);
        s(7) <= (8.0,0.0);
     wait;                                  --WAIT STATEMENT
   end process;                             --END OF PROCESS
end behavior;                               --END OF ARCHITECTURE
```

4.2 IDFT Using DIF FFT

4.2.1 Package

```vhdl
library IEEE;                                               --IMPORTING LIBRARY
use IEEE.std_logic_1164.all;
use IEEE.MATH_REAL.ALL;                                     --USING MATH_REAL LIBRARY
-----------------------------------------------------------------------------------------------
package dif_ifft_pkg is                                     --PACKAGE DECLARATION
type complex is                                             --DEFINING A DATA STRUCTURE
    record                                                  --DEFINING RECORD
        r : real;
        i : real;
    end record;

--DECALRING AN ARRAY OF TYPE COMPLEX OF LENGTH = 8
type ar is array (0 to 7) of complex;
--DECLARING AN ARRAY OF TYPE COMPLEX OF LENGTH = 4
type ar2 is array (0 to 3) of complex;

--FUNCTION DECLARATION OF ADDITION
function add (n1,n2 : complex) return complex;
--FUNCTION DECLARATION OF SUBSTRACTION
function sub (n1,n2 : complex) return complex;
--FUNCTION DECLARATION OF MULTIPLICATION
function multi (n1,n2 : complex) return complex;

end dif_ifft_pkg;
-----------------------------------------------------------------------------------------------
package body dif_ifft_pkg is                                --START OF PACKAGE BODY
-----------------------------------------------------------------------------------------------
--FUNCTION FOR ADDITION
function add (n1,n2 : complex) return complex is
variable s : complex;                                       --VARIABLE DECLARATION
begin
s.r:=n1.r + n2.r;                                         --ADDITION OF REAL PARTS
s.i:=n1.i + n2.i;                                         --ADDITOIN OF IMAGINARY PARTS
return s;                                                 --RETURNING SUM
end add;
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
--FUNCTION FOR SUBSTRACTION
function sub (n1,n2 : complex) return complex is
variable d : complex;                                     --VARIABLE DECLARATION
begin
d.r:=n1.r - n2.r;                                         --SUBSTRACTING REAL PARTS
d.i:=n1.i - n2.i;                                         --SUBSTRACTING IMAGINARY PARTS
return d;                                                 --RETURNING SUBSTRACTED VALUE
end sub;
-----------------------------------------------------------------------------------------------
--FUNCTION FOR MULTIPLICATION.
function multi (n1,n2 : complex) return complex is
```

```
variable p : complex;                          --VARIABLE DECLARATION
begin
p.r:=(n1.r * n2.r) - (n1.i * n2.i);            --MULTIPLYING REAL WITH ...
    IMAGINARY VALUE
p.i:=(n1.r * n2.i) + (n1.i * n2.r);            --MULTIPLYING IMAGINARY PARTS
return p;                                      --RETURNING MULTIPLIED VALUE
end multi;
-----------------------------------------------------------------------------
end dif_ifft_pkg;                              --END OF PACKAGE BODY
```

4.2.2 Main Code

```
library IEEE;                                  --IMPORTING LIBRARY
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.MATH_REAL.ALL;
library work;                                  --USING FILES FROM WORK ...
    DIRECTORY
use work.dif_ifft_pkg.ALL;                     --USING PACKAGE ...
    DIF_IFFT_PKG FROM WORK DIRECTORY
-----------------------------------------------------------------------------
entity dif_ifft_8pt is                         --ENTITY DECLARATION
port(   s : in ar;                             --INPUT SIGNALS
        y : out ar);                           --OUTPUT SIGNALS
end dif_ifft_8pt;
-----------------------------------------------------------------------------
architecture Behavioral of dif_ifft_8pt is     --ARCHITECTURE DECLARATION
component butterfly is                         --COMPONENT DECLARATION
   port(
      b1,b2 : in complex;                      --INPUTS OF BUTTERFLY ...
          STRUCTURE
      w :in complex;                           --PHASE FACTOR
      z1,z2 :out complex);                     --OUTPUTS OF BUTTERFLY ...
          STRUCTURE
end component;
----------------------------------------------------------------------------
--DEFINING SIGNLAS Z1 AND Z2 WITH DEFAULT VALUE OF(0.0,0.0)
signal z1,z2 : ar ;

--PHASE FACTOR, w_N = e^(-j*(2*p/N)*n)
--w_N^n = cos((2*p/N)*n) - j*sin((2*p/N)*n);
--w_N VALUES FOR n = 0-4
constant w : ar2 := ( (1.0,0.0), (0.7071,0.7071), (0.0,1.0), ...
    (-0.7071,0.7071) );
signal p:ar;                                  --DEFINING SIGNAL P OF TYPE AR
begin
--FIRST STAGE OF BUTTERFLY, n = 0,2,4,6; N = 8
--MAPPING INPUTS AND OUTPUTS
bfly1 : butterfly port map(s(0),s(4),w(0),z1(0),z1(1));
bfly2 : butterfly port map(s(2),s(6),w(2),z1(2),z1(3));
bfly3 : butterfly port map(s(1),s(5),w(1),z1(4),z1(5));
bfly4 : butterfly port map(s(3),s(7),w(3),z1(6),z1(7));
```

```
--SECOND STAGE OF BUTTERFLY, n = 0,2; N = 8
--MAPPING INPUTS AND OUTPUTS
bfly5 : butterfly port map(z1(0),z1(2),w(0),z2(0),z2(2));
bfly6 : butterfly port map(z1(1),z1(3),w(0),z2(1),z2(3));
bfly7 : butterfly port map(z1(4),z1(6),w(2),z2(4),z2(6));
bfly8 : butterfly port map(z1(5),z1(7),w(2),z2(5),z2(7));

--THIRD STAGE OF BUTTERFLY, n = 0; N = 8
--MAPPING INPUTS AND OUTPUT
bfly9 : butterfly port map(z2(0),z2(4),w(0),p(0),p(4));
bfly10 : butterfly port map(z2(1),z2(5),w(0),p(1),p(5));
bfly11 : butterfly port map(z2(2),z2(6),w(0),p(2),p(6));
bfly12 : butterfly port map(z2(3),z2(7),w(0),p(3),p(7));

y(0)<= multi(p(0),(0.125,0.0));           --DIVIDING BY FACTOR OF 1/N
y(1)<= multi(p(7),(0.125,0.0));           --AND ASSIGNING TO OUTPUT SIGNAL
y(2)<= multi(p(6),(0.125,0.0));
y(3)<= multi(p(5),(0.125,0.0));
y(4)<= multi(p(4),(0.125,0.0));
y(5)<= multi(p(3),(0.125,0.0));
y(6)<= multi(p(2),(0.125,0.0));
y(7)<= multi(p(1),(0.125,0.0));
end Behavioral;
```

4.2.3 Butterfly Structure

```
library IEEE;                                 --IMPORTING LIBRARY
use IEEE.STD_LOGIC_1164.ALL;
library work;                                 --USING FILES FROM WORK ...
   DIRECTORY
use work.dif_ifft_pkg.ALL;                    --USING PACKAGE ...
   DIF_IFFT_PKG FROM WORK DIRECTORY
--------------------------------------------------------------------------
entity butterfly is                           --ENTITY DECLARATION
   port(
      b1,b2 : in complex;                     --INPUTS OF BUTTERFLY ...
         STRUCTURE
      w :in complex;                          --PHASE FACTOR
      z1,z2 :out complex);                    --OUTPUTS OF LIBRARY
end butterfly;
--------------------------------------------------------------------------
architecture Behavioral of butterfly is       --ARCHITECTURE DECLARATION
signal z2_temp : complex;                     --SIGNAL DECLARATION

begin
z1 <= add(b1,b2);                       --BUTTERFLY EQUATION FOR ADDITION
z2_temp <= sub(b1,b2);
z2 <= multi(z2_temp,w);                 --BUTTERFLY EQUATION FOR ...
   SUBSTRACTION
end Behavioral;
```

4.2.4 Testbench

```
library ieee;                                        --IMPORTING LIBRARY
use ieee.std_logic_1164.all;
library work;                                        --WORKING IN CURRENT(WORK) ...
    FOLDER
use work.dif_ifft_pkg.all;                           --USING PACKAGE ...
    DIF_IFFT_PKG FROM WORK DIRECTORY
-----------------------------------------------------------------------------------
entity tb is                                         --TESTBENCH ENTITY DECLARATION
end tb;
-----------------------------------------------------------------------------------
architecture behavior of tb is                       --ARCHITECTURE DECLARATION
signal s,y : ar;                                     --SIGNAL DECLARATION
begin
-- Instantiating the Unit Under Test (UUT)
uut: entity work.dif_ifft_8pt port map (s => s,y => y);
   stim_proc: process                                -- Stimulus process
   begin
        s(0) <= (1.0,0.0);                           --GIVING INPUTS
        s(1) <= (2.0,0.0);
        s(2) <= (3.0,0.0);
        s(3) <= (4.0,0.0);
        s(4) <= (5.0,0.0);
        s(5) <= (6.0,0.0);
        s(6) <= (7.0,0.0);
        s(7) <= (8.0,0.0);
     wait;                                           --WAIT STATEMENT
   end process;                                      --END OF PROCESS
end behavior;                                        --END OF ARCHITECTURE
```

5 Code Explanation

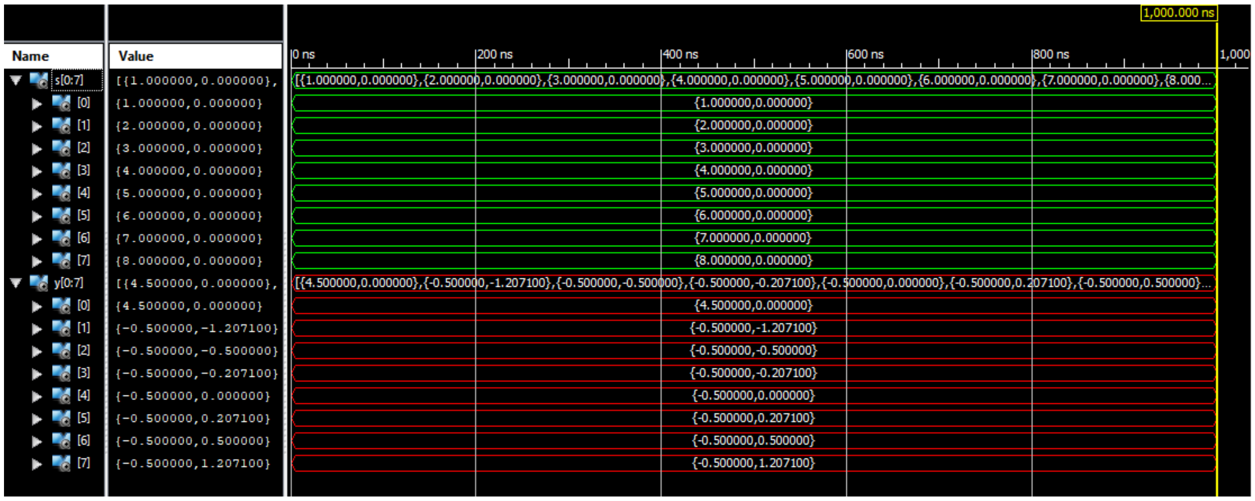
5.1 Package

A packaged declaration has been done in which the different functions, data types, arrays etc are defined which will be used in the program of FFT. For taking input as complex, we have imported the MATH-REAL package. In VHDL, a composite signal (record or array) is viewed as the juxtaposition* of its scalar sub-elements acting as individual signals. So the driving value and effective value of the composite signal is defined in terms of the driving value and effective values of its scalar sub-elements. From the definition mentioned in the package, s, d, and p are composite signals formed by juxtaposition of the two signals each respectively ((s.r,s.i),(d.r,d.i),(p.r,p.i)). And hence, the input signal (input value) might be a complex number which is combination of real and imaginary parts, we have written it as composite signal. An array to store such complex values of input, output and W_N is made of type complex. Inside package body, the function definition are written for addition, subtractions and multiplication involved in the algorithm for such composite inputs (Complex inputs).

5.2 Main code

When we run the code, the functions and other data types declared in the package are called from the package. While giving inputs, we have to take care that the data type of the inputs and outputs declared inside the entity are of the array type which we defined in the package. Now, as we call the Butterfly file In the main code, values are mapped to the inputs and outputs of butterfly vhd file. In the formula of DFT we came across W_N , so we also need to define those values inside the main code and hence the values given in the code. As soon as the butterfly file is called, the mapped values are then again passed to the addition, subtraction, and multiplication function called inside the architecture. You may find that those functions(eg. multi()) has been called in the main code to multiply the 1/N factor to the output.

6 Observation



```
Name      Value
s[0:7]    [{1.000000,0.000000},
  [0]     {1.000000,0.000000}
  [1]     {2.000000,0.000000}
  [2]     {3.000000,0.000000}
  [3]     {4.000000,0.000000}
  [4]     {5.000000,0.000000}
  [5]     {6.000000,0.000000}
  [6]     {7.000000,0.000000}
  [7]     {8.000000,0.000000}
y[0:7]    [{4.500000,0.000000},
  [0]     {4.500000,0.000000}
  [1]     {-0.500000,-1.207100}
  [2]     {-0.500000,-0.500000}
  [3]     {-0.500000,-0.207100}
  [4]     {-0.500000,0.000000}
  [5]     {-0.500000,0.207100}
  [6]     {-0.500000,0.500000}
  [7]     {-0.500000,1.207100}
```

```
[{1.000000,0.000000},{2.000000,0.000000},{3.000000,0.000000},{4.000000,0.000000},{5.000000,0.000000},{6.000000,0.000000},{7.000000,0.000000},{8.000...
[{4.500000,0.000000},{-0.500000,-1.207100},{-0.500000,-0.500000},{-0.500000,-0.207100},{-0.500000,0.000000},{-0.500000,0.207100},{-0.500000,0.500000}...
```

Figure 4: Output

The IDFT are obtained and the RTL Schematic is not possible so it can not be implemented on FPGA board.

7 Conclusion

The simulation results show that IDFT was successfully implemented on Xilinx ISE Software by both DIT-FFT and DIF-FFT algorithms. The problem faced was that the efficient 8-point IDFT architecture proposed in this project could not be synthesized and implemented on FPGA Board because of the absence of sensitivity list and the wait statement. The absence of the sensitivity list implies to the fixed values of the inputs and so the wait statement came to picture.

8 References

1)https://scialert.net/fulltextmobile/?doi=itj.2012.118.125
2)Design and VHDL Implementation of 6-Point FFT 8-Point FFT/IFFT by Ramanbhai Patil
3)Implementation of IDFT algorithm paper by M. Verderber A. Zemva
4)Efficient Implementation of 64-Point FFT/IFFT for OFDM on FPGA paper by Mr. Shreyas D. Deshmukh Mrs. Deepali Sale