1. Implementation of IFFT

Inverse Fast Fourier Transform can be calculated using the same architecture used to calculate FFT. This is done by swapping the real and imaginary part of the primary inputs and performing FFT on these values and then swapping the real and imaginary part of the outputs and dividing it by 8. The architecture now has another input called 'mode' whose value determines whether to perform FFT(mode=0) or IFFT(mode=1). In the architecture, this is done using two 2:1 multiplexers which have mode as their control signal. The real part and imaginary part of input are given to both these multiplexers. The output of the 1st mux feeds the real part of the input to the corresponding input wire of the architecture, and the output of the 2nd mux feeds the imaginary part of the input to the corresponding input wire of the architecture. A similar procedure is done to exchange the real and imaginary parts of the output values.

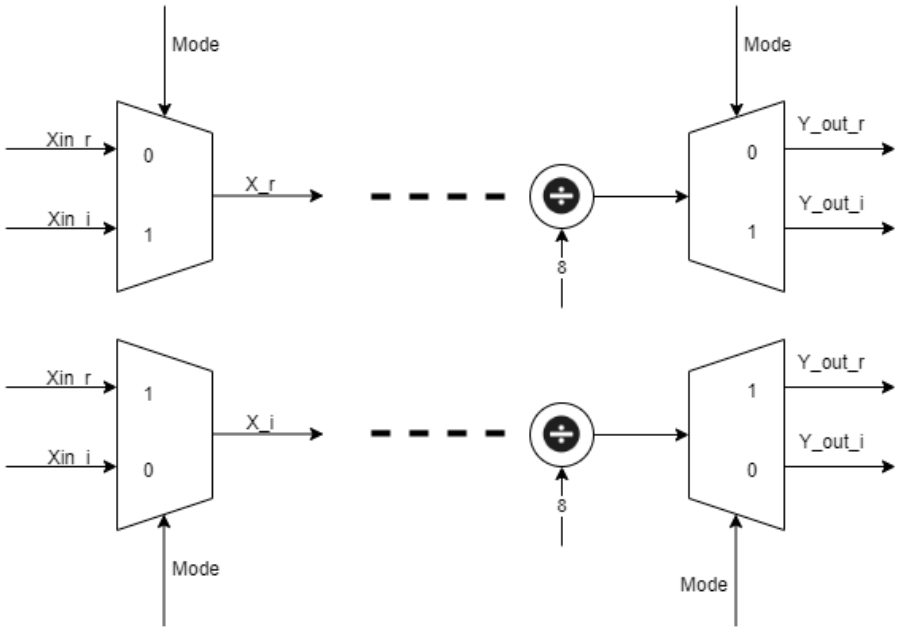

Fig : Exchanging the real and imaginary parts of inputs and outputs according to the control signal 'mode'

2. Experimental Setup

The testbench designed for the initial FFT/IFFT design has been reused with the same inputs. The Matlab code for testing remains the same. The mode has been set to 1 for IFFT calculation.

3. Experimental Results

Matlab code for displaying IFFT:

```
in=[0.5,-1,-1,-1,-1,-1,-1,-1,
   -1,0.5,-1,-1,-1,-1,-1,-1,
   -1,-1,0.5,-1,-1,-1,-1,-1,
   -1,-1,-1,0.5,-1,-1,-1,-1,
   -1,-1,-1,-1,0.5,-1,-1,-1,
   -1,-1,-1,-1,-1,0.5,-1,-1,
   -1,-1,-1,-1,-1,-1,0.5,-1,
   -1,-1,-1,-1,-1,-1,-1,0.5];
X=ifft(in);
disp((X));
```

Matlab output(each set of 8 inputs as different columns):

-0.8125 + 0.0000i	-0.8125 + 0.0000i	-0.8125 + 0.0000i	-0.8125 + 0.0000i	-0.8125 + 0.0000i	-0.8125 + 0.0000i	-0.8125 + 0.0000i	-0.8125 + 0.0000i
0.1875 + 0.0000i	0.1326 + 0.1326i	0.0000 + 0.1875i	-0.1326 + 0.1326i	-0.1875 + 0.0000i	-0.1326 - 0.1326i	0.0000 - 0.1875i	0.1326 - 0.1326i
0.1875 + 0.0000i	0.0000 + 0.1875i	-0.1875 + 0.0000i	0.0000 - 0.1875i	0.1875 + 0.0000i	0.0000 + 0.1875i	-0.1875 + 0.0000i	0.0000 - 0.1875i
0.1875 + 0.0000i	-0.1326 + 0.1326i	0.0000 - 0.1875i	0.1326 + 0.1326i	-0.1875 + 0.0000i	0.1326 - 0.1326i	0.0000 + 0.1875i	-0.1326 - 0.1326i
0.1875 + 0.0000i	-0.1875 + 0.0000i	0.1875 + 0.0000i	-0.1875 + 0.0000i	0.1875 + 0.0000i	-0.1875 + 0.0000i	0.1875 + 0.0000i	-0.1875 + 0.0000i
0.1875 + 0.0000i	-0.1326 - 0.1326i	0.0000 + 0.1875i	0.1326 - 0.1326i	-0.1875 + 0.0000i	0.1326 + 0.1326i	0.0000 - 0.1875i	-0.1326 + 0.1326i
0.1875 + 0.0000i	0.0000 - 0.1875i	-0.1875 + 0.0000i	0.0000 + 0.1875i	0.1875 + 0.0000i	0.0000 - 0.1875i	-0.1875 + 0.0000i	0.0000 + 0.1875i
0.1875 + 0.0000i	0.1326 - 0.1326i	0.0000 - 0.1875i	-0.1326 - 0.1326i	-0.1875 + 0.0000i	-0.1326 + 0.1326i	0.0000 + 0.1875i	0.1326 + 0.1326i

Real Part of outputs from Verilog:

-0.8125
0.1875
0.1875
0.1875
0.1875
0.1875
0.1875
0.1875
-0.8125
0.132568359375
0
-0.132568359375
-0.1875
-0.132568359375
0
0.132568359375
-0.8125
0
-0.1875
0
0.1875
0
-0.1875
0
-0.8125
-0.132568359375
0
0.132568359375
-0.1875
0.132568359375
0
-0.132568359375
-0.8125
-0.187255859375
0.187255859375
-0.187255859375
0.187255859375
-0.187255859375
0.187255859375
-0.187255859375
-0.8125

-0.132568359375
0
0.13232421875
-0.187255859375
0.132568359375
0
-0.13232421875
-0.8125
0
-0.187255859375
0
0.187255859375
0
-0.187255859375
0
-0.8125
0.13232421875
0
-0.132568359375
-0.187255859375
-0.13232421875
0
0.132568359375

Imaginary Part of outputs from Verilog:
0
0
0
0
0
0
0
0
0.132568359375
0.1875
0.132568359375
0
-0.132568359375
-0.1875
-0.132568359375
0
0.1875
0
-0.1875
0
0.1875
0
-0.1875

0
0.132568359375
-0.1875
0.132568359375
0
-0.132568359375
0.1875
-0.132568359375
0
0
0
0
0
0
0
0
0
-0.132568359375
0.187255859375
-0.132568359375
0
0.132568359375
-0.187255859375
0.132568359375
0
-0.187255859375
0
0.187255859375
0
-0.187255859375
0
0.187255859375
0
-0.132568359375
-0.187255859375
-0.132568359375
0
0.132568359375
0.187255859375
0.132568359375

The outputs can be seen to be the same verifying our results