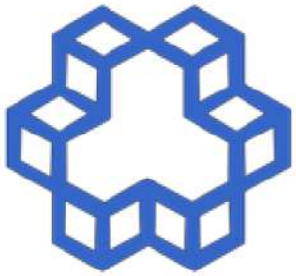


K. N. Toosi
University of Technology

High pass FIR (16 TAP) implementation

Done by:

Mehran Tamjidi

Note: the filter coefficients and input data are supposed to be 10 and 8 bit respectively

Table of Contents

Direct form of FIR filter implementation

Theoretical filter designing

The Direct form implementation for FIR filters is the most convenient method consisting of filter coefficients multiplied by the input signal reaching the output after a delay. Each column models one tap.

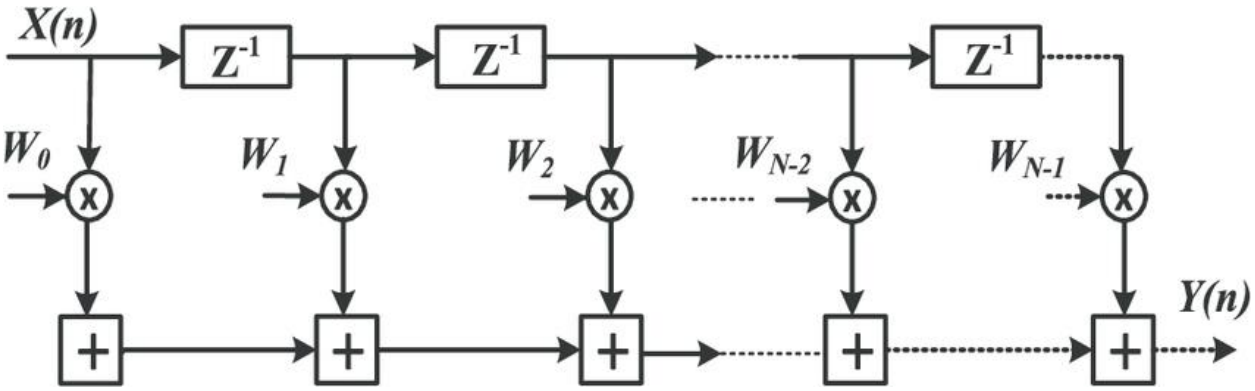


Fig1- Direct form implementation of fir filter

The first theological part is designing the corresponding FIR filter with the MATLAB filter designer toolbox.

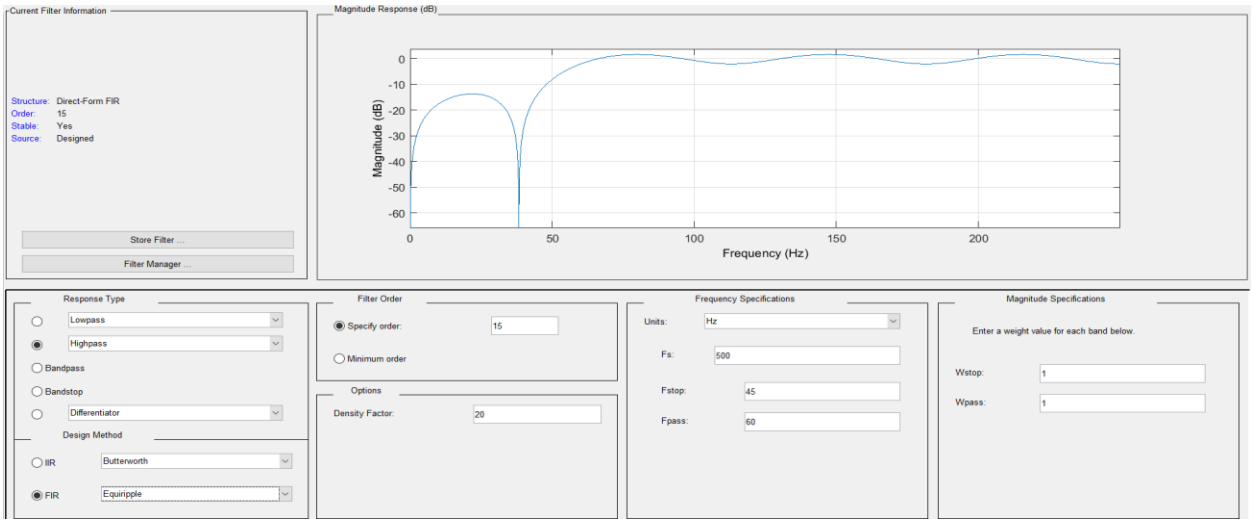


Fig2 -filter designer scheme and the imported values

Due to the high pass implementation, the filter should pass 70 Hz frequency (determined in the attached file). This frequency can be passed by a stop frequency of 45 and a pass frequency of 60 Hz. The sampling frequency is selected 500 specified in the question. The 16-tap FIR filter needs 16 multipliers, which means the order is 15.

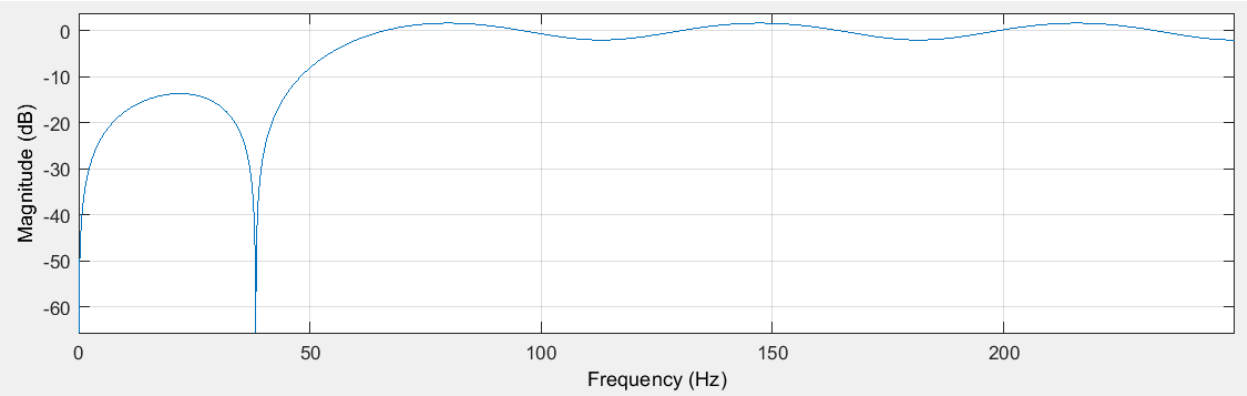


Fig3 -filter magnitude

The filter parameters obtained as below:

```
Discrete-Time FIR Filter (real)
-------------------------------
Filter Structure  : Direct-Form FIR
Filter Length     : 16
Stable            : Yes
Linear Phase      : Yes (Type 4)

Implementation Cost
Number of Multipliers            : 16
Number of Adders                 : 15
Number of States                 : 15
Multiplications per Input Sample : 16
Additions per Input Sample       : 15
```

Fig4 -filter description

```
Numerator:
-0.068073423429381968441376216105709318072
 0.071849400790840914354795643248507985845
 0.078029079866443529223118957816041074693
 0.083673354969798038882622392975463299081
 0.068525060029565348918012546164391096681
 0.013822585865391756598796746402513235807
-0.114579157439376050353629921119136270136
-0.601724422211067477839208095247158780694
 0.601724422211067477839208095247158780694
 0.114579157439376050353629921119136270136
-0.013822585865391756598796746402513235807
-0.068525060029565348918012546164391096681
-0.083673354969798038882622392975463299081
-0.078029079866443529223118957816041074693
-0.071849400790840914354795643248507985845
 0.068073423429381968441376216105709318072
```

Fig5 -filter coefficients

Thanks to the filter designer toolbox, the filter coefficients would directly export to the Workspace, and with the below code they can be converted to binary and can be used in VHTL code.

```
b16_1024=fix(511/max(abs(b16)).*b16); %10 bit representation of filter coefficients
```

As the filter coefficients are 10-bit, this code has changed with respect to the attached .m file format (it was 8-bit).

The FFT (shift) analyses of the filtered sum of these three inputs are illustrated in figure 6.

```
x1(k)=sin(2*pi*f1*k/fs);
x2(k)=cos(2*pi*f2*k/fs);
x3(k)=sin(2*pi*f3*k/fs);
```

input =`x1+x2+x3;`

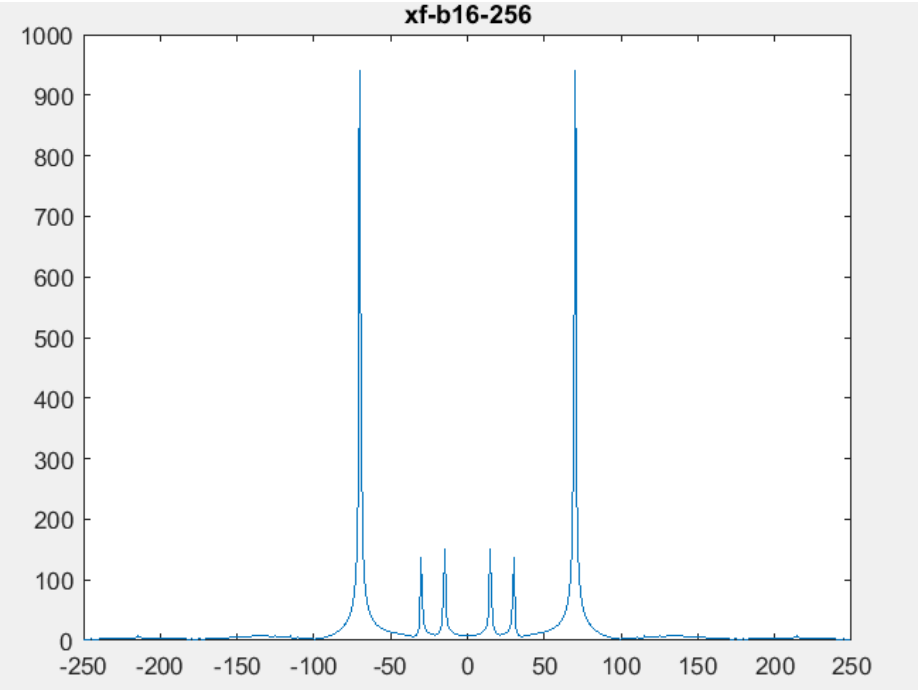


Fig6 -FFT analyses of filtered input

As shown in figure 6 the 70 Hz is filtered meaning that our filter theoretical implementation has worked.

Figure 7 analyzes its precision by zooming in, in the vicinity of 70 Hz.

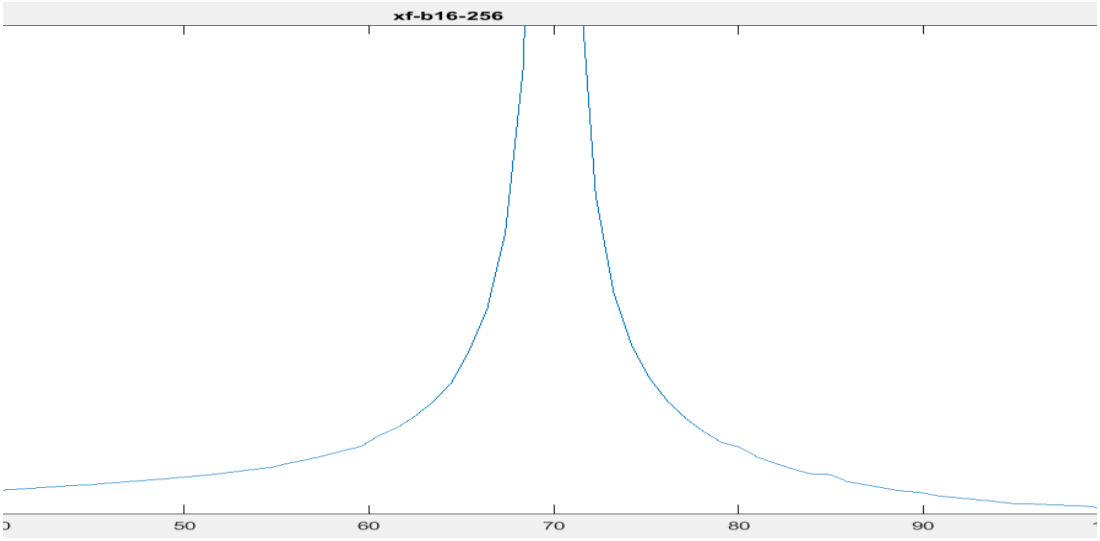
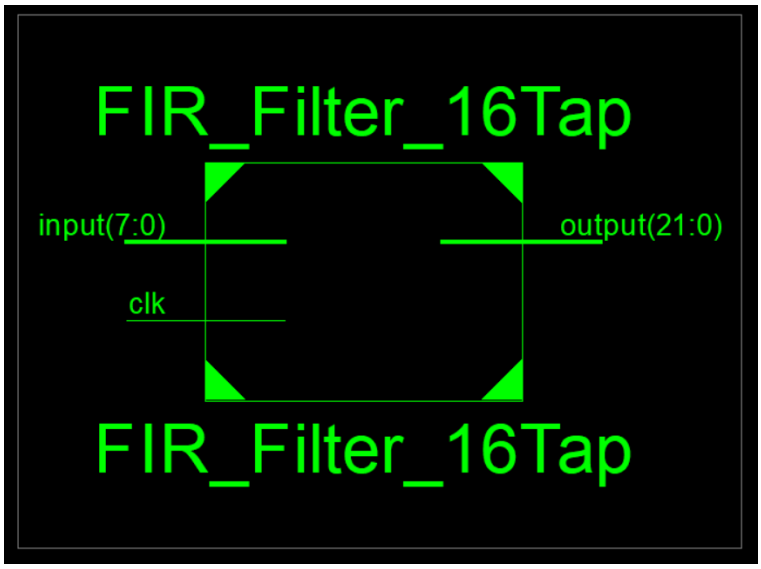


Fig7 -FFT analyses of filtered input with zooming in

implementation with VHDL

The more detailed are as the same was mentioned in the class.

The input data considered to be 8 bit and the output is 22-bit because there is multiplication between the 8-bit input data and the 10-bit coefficient and also 4 step summation can produce 22 bits as output. The schematic is as below.



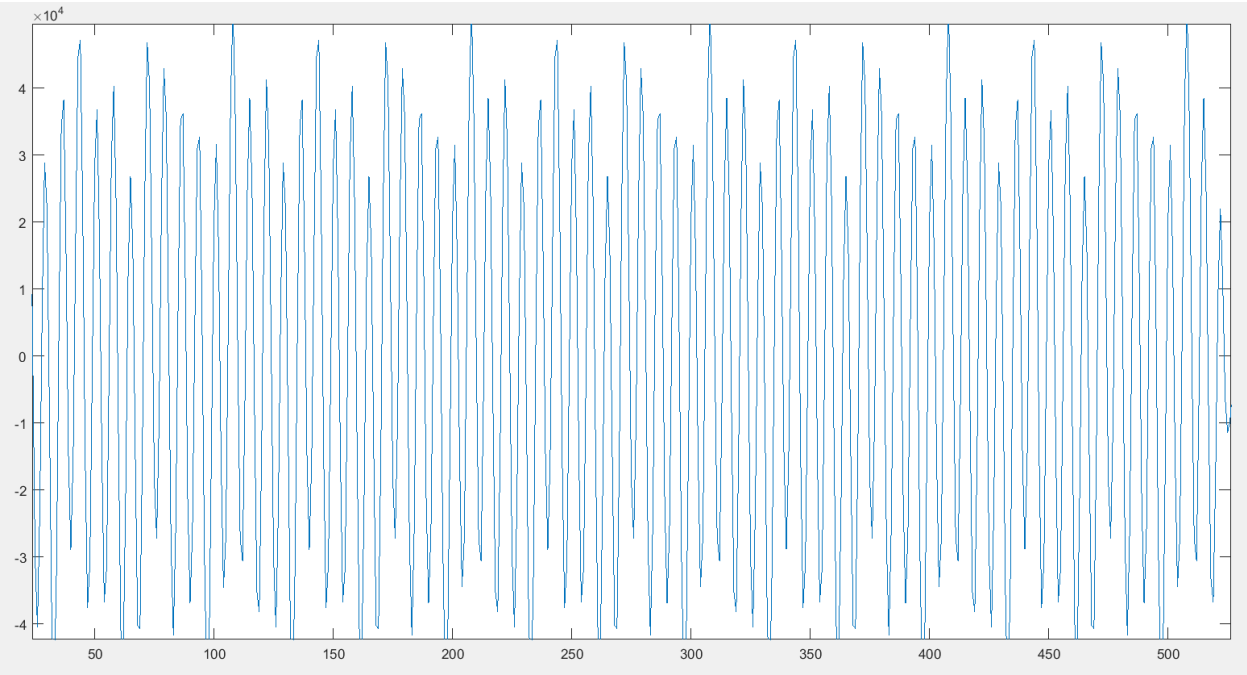
We use the antisymmetric properties of FIR filter coefficients, which means that all the only half of the coefficients are calculated and entered to the signal processing part of VHDL code.

Because in a high pass filter all the coefficients are antisymmetric and we can use this properties ,we use minus instead of the summation in the block diagram. But instead, we use half of the symmetric coefficients.

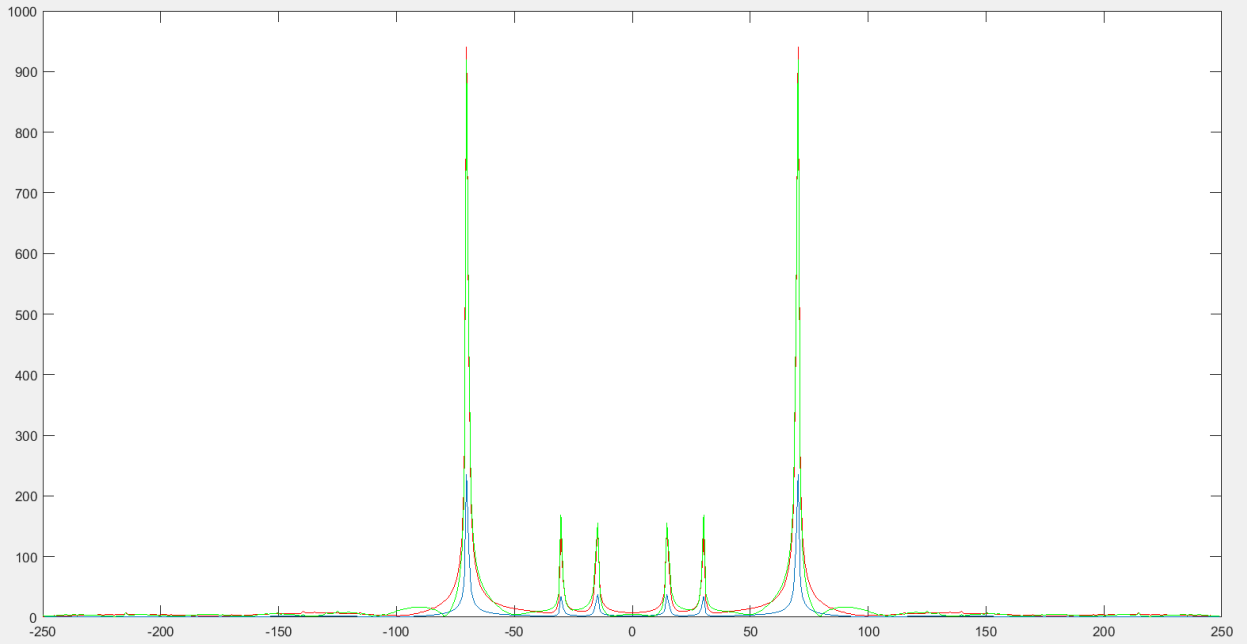
Empirical results

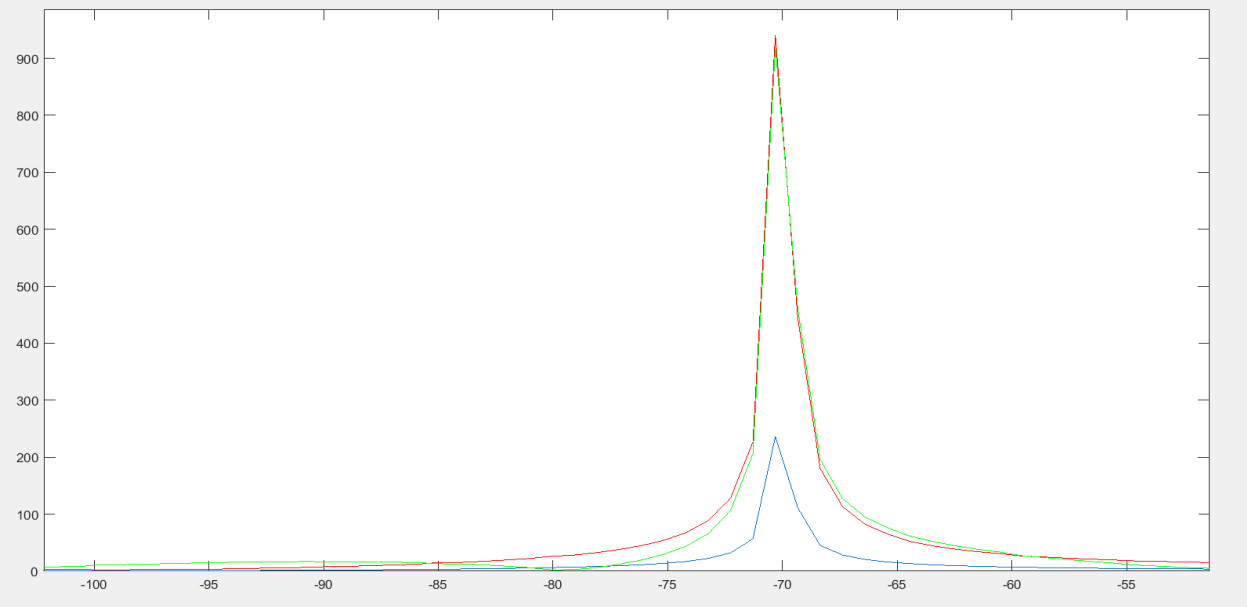
Given the results which have been saved in output.txt placed in repository to MATLAB manually, will provide visual comparation between theoretical and empirical results.

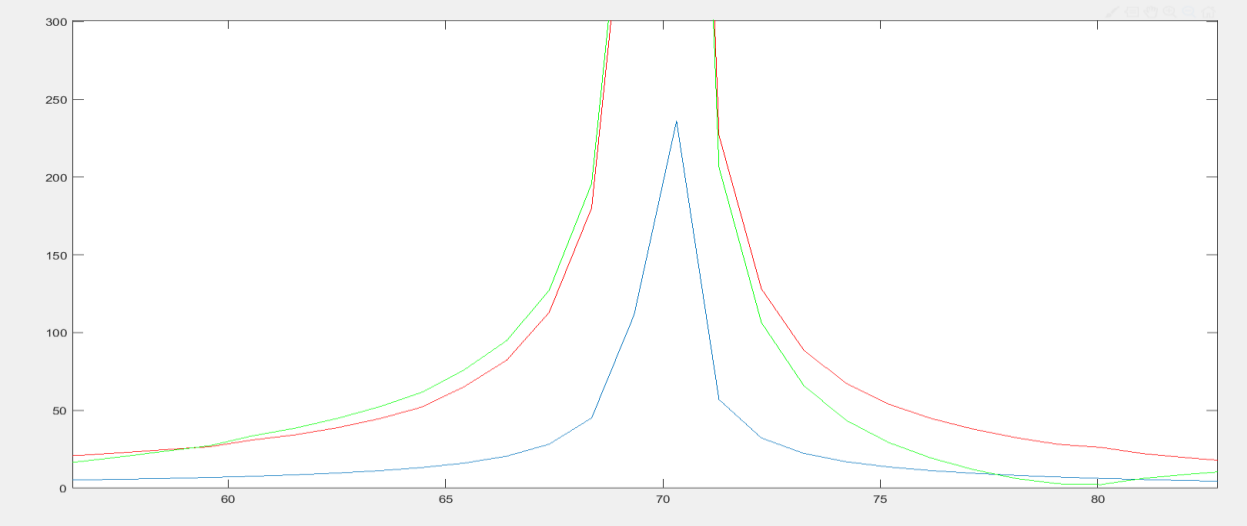
Plotted output is as below.



As illustrated below, the green frequency magnitude indicates the theoretical results and the red color is the hardware implementation one has coincided.


1000
900
800
700
600
500
400
300
200
100
0
-250
-200
-150
-100
-50
0
50
100
150
200
250


900
800
700
600
500
400
300
200
100
0
-100
-95
-90
-85
-80
-75
-70
-65
-60
-55



By sending output.txt entities in to the variable output and y (in an array) in MATLAB and appending to the .m file in the attached file. this comparation will be resulted.

Some command must be run to send output.txt contents to the variable output and y and plot this.

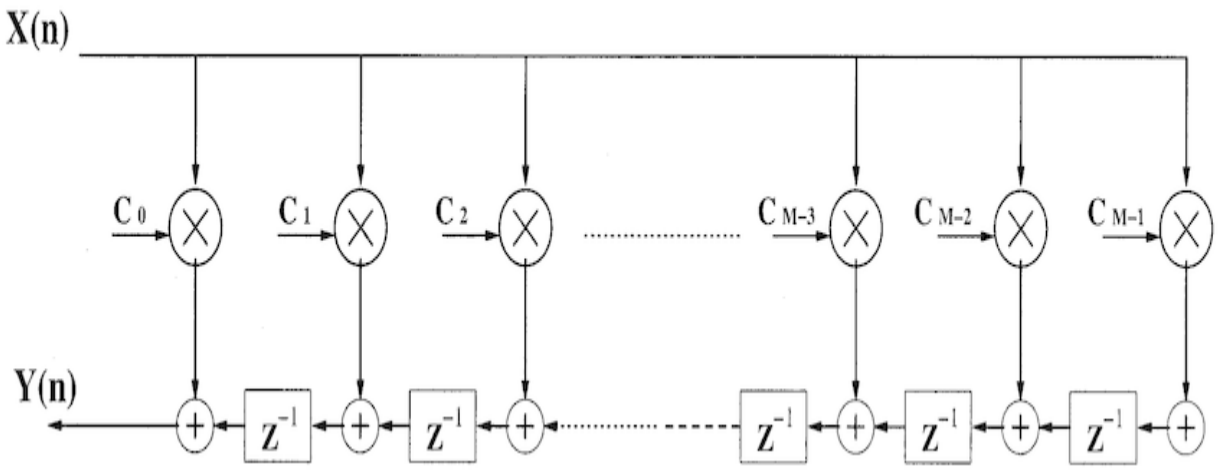
```
>> x = 1:100;
>> plot(x,output)
>> y = output;
```

The result shows that both empirical and theoretical results are coincided to each other and which is a good specification of our hardware implementation.

Transpose form of FIR filter

Theoretical filter designing

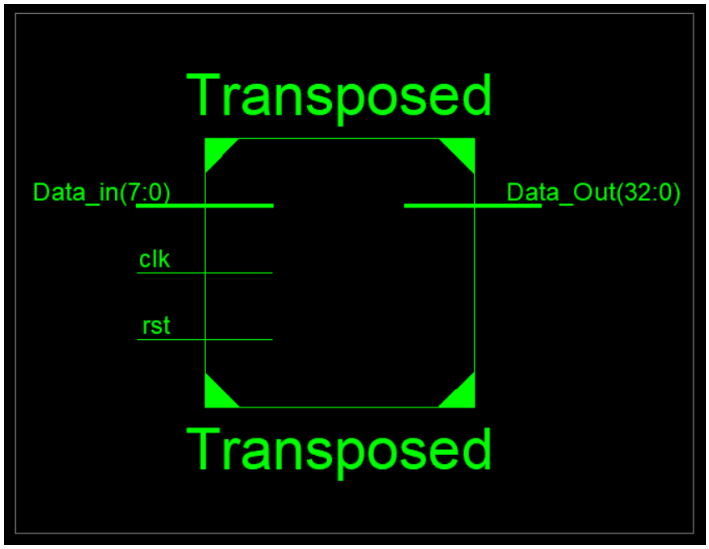
For the transposed form the input and output in the previous scheme are replaced and the delay placed is also reversed.



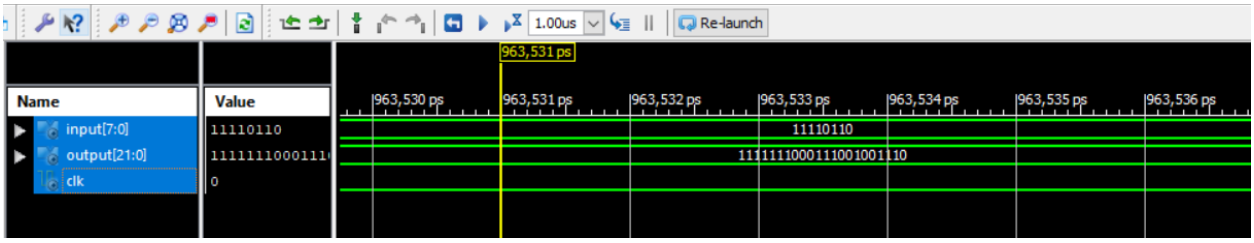
implementation with VHDL

The more detailed are as the same was mentioned in the class and .

The input data considered to be 8 bit and the output is 33-bit because there is multiplication between the 8-bit input data and the 10-bit coefficient with each step one summation can generate one more bit. After all final output would be 33bit.

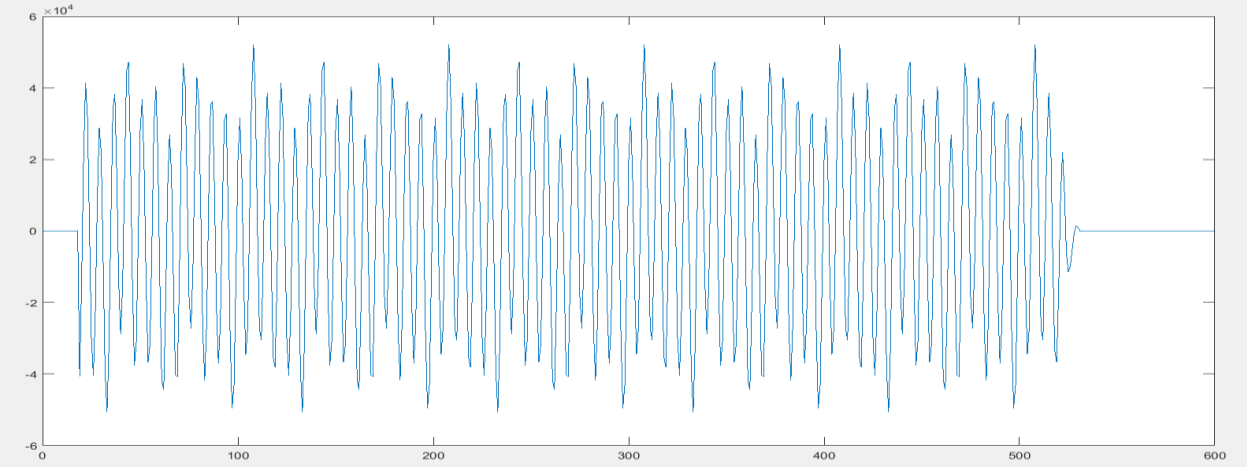


The output is like the previous part was send to matlab

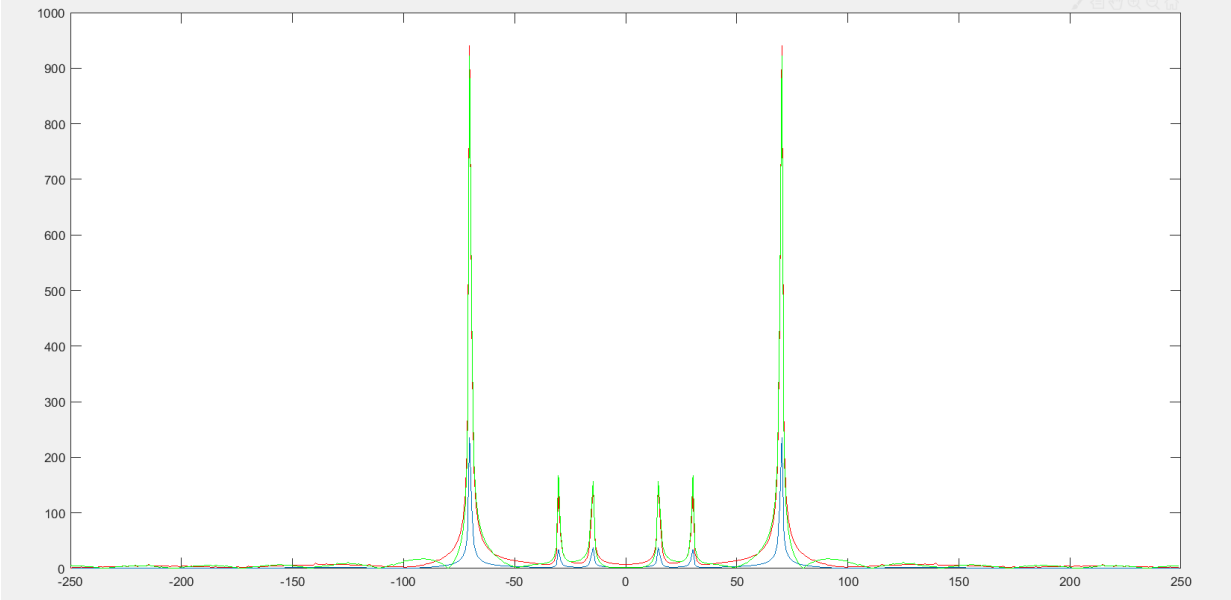


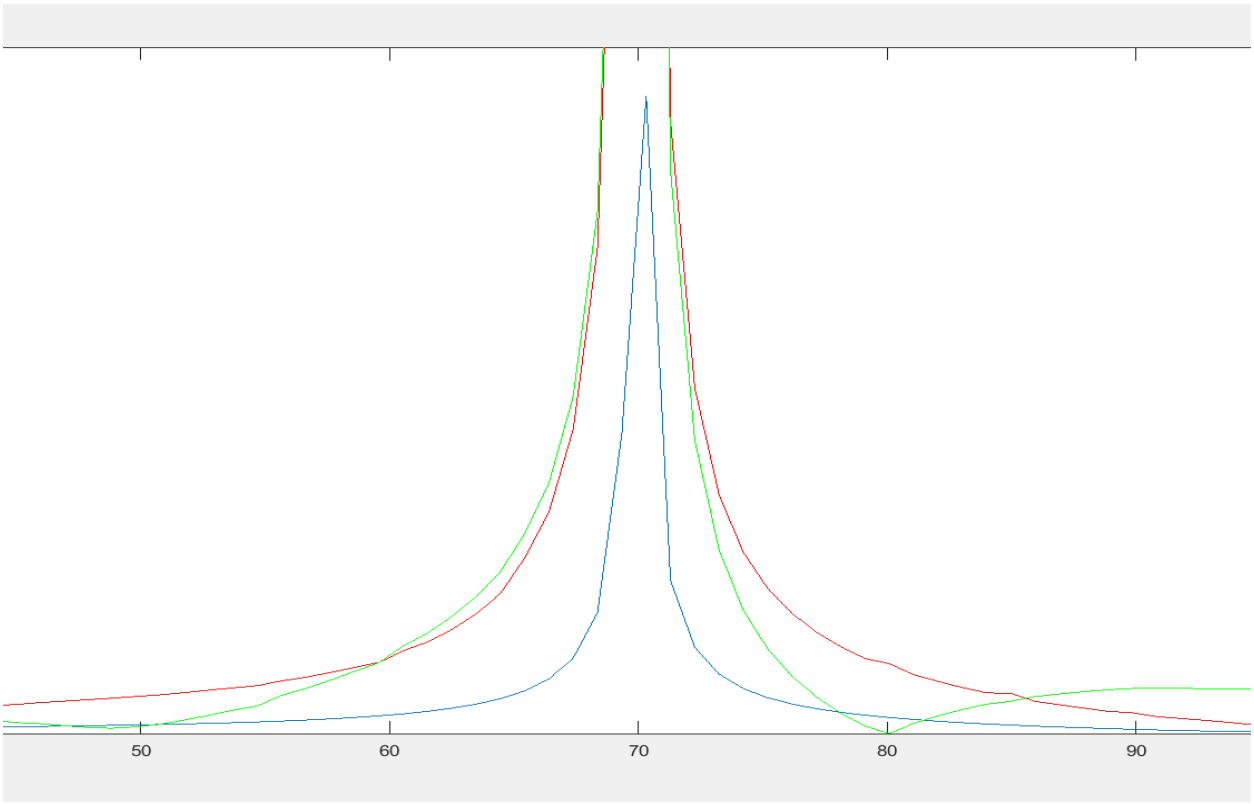
Empirical results

The hardware implemented parameters in Xilinx ISE are sanded to MATLAB and the output is as below.

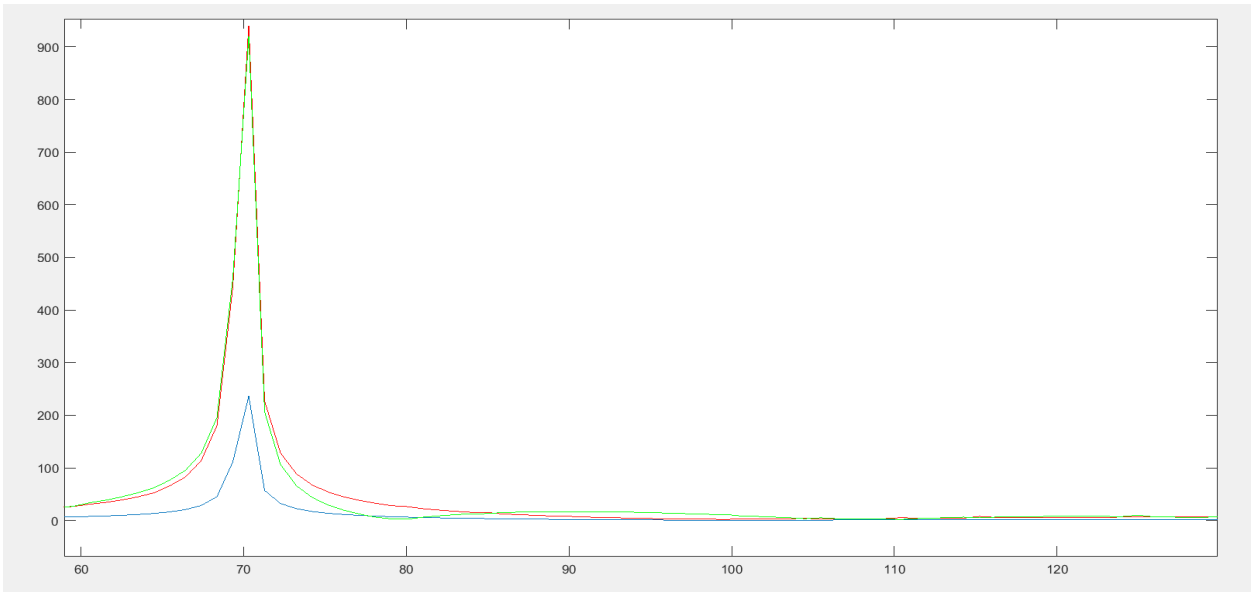


Like the previous part the FFT analysis from different scales are demonstrated as below.





The designed filter is acceptable and like the non-transposed form the two empirical and theoretical spectrum have very little differences.



Maximum frequency comparation between two methods

The maximum frequency for the second state implementation was 327.04MHz which is good speed. And we reached higher speed and maximum frequency by using transform model of fir filter with respect to the first model we implemented.

So, transposing can increase maximum frequency.

For the first situation

```
Minimum period: 4.349ns (Maximum Frequency: 229.959MHz)
Minimum input arrival time before clock: 3.227ns
Maximum output required time after clock: 0.643ns
```

And for the secend situation(trasposed)

```
Minimum period: 3.058ns (Maximum Frequency: 327.045MHz)
Minimum input arrival time before clock: 2.674ns
Maximum output required time after clock: 2.300ns
Maximum combinational path delay: No path found
```