International Journal of Research in Engineering, Science and Management
Volume 5, Issue 12, December 2022
https://www.ijresm.com | ISSN (Online): 2581-5792

Case Teaching of MATLAB Implementation of FIR Filter with an Overview of Filter Analogies

Lolla Mahidhar[1*], Apurva Kumari[2]
[1]Student, Department of Electronics and Communication Engineering, B. V. Raju Institute of Technology, Medak, India
[2]Associate Professor, Department of Electronics and Communication Engineering, B. V. Raju Institute of Technology, Medak, India

***Abstract*: FIR Filters have been a key component of digital signal processing's work for a very long time. These filters have expanded in significance in terms of the signal processing with the introduction of newer technology. Although numerous models of the FIR Filter have been built by researchers using a variety of techniques, some potential problems that should be focused on during these implementations are only addressed. Therefore, the work here ensures that a number of those problems are addressed during the implementation over MATLAB and that solutions are provided in order to greatly increase the effectiveness of the implementation. The misconceptions about magnitude and input/output responses, the idea of group delay, and difficulties with direct convolutional implementation are some examples. Additionally, several analogies are given to explain the FIR Filter in terms of its mathematical models and also the use of the weighted average filter is illuminated.**

***Keywords*: MATLAB, FIR filter, Group delay, Magnitude response.**

1. Introduction

In the past few decades, a variety of adaptable designs for the suggested FIR filter design in signal processing have already been introduced, culminating in discussions and implementations of its use in numerous applications. Nevertheless, there has been a lot more development and study in the design and application of the suggested filter for the correct applications. One of these great improvements is the modification of the development board utilised for the FIR filter's implementation.

According to the literature, the first FIR filter designs developed roughly ten years ago used basic adders and multipliers from digital electronics. They also included the use of sequential devices like flip flops to create the necessary delay for the filter equipment to operate effectively. The appropriate Verilog code for its execution was synthesised onto different FPGA boards, including the Spartan 6, Xilinx Zynq-7000 Soc, and Xilinx Virtex 6 FPGA device. In order to comprehend how adaptable an FPGA Board is in the implementation of such DSP-based high computational circuits; this study was conducted.

As a result, new developments in architecture are now possible. Flip-flop models were suggested to be replaced by buffers, signed multipliers were taken into consideration, and enhancements to adder circuits were also suggested. Significant issues like noise, interference, delay, and high resource use emerged as the filter's order was raised. The data path and control path-based architecture that will be proposed will therefore put a strong emphasis on minimising delay and resource usage.

In particular, the filer implementation in digital signal processing has traditionally used MATLAB. The study explicitly outlines MATLAB's use cases for the filter implementation in addition to the standard implementation. This may take into account elements like coefficient generation, delay (phase or group), impulse responses, etc. The development of HDL has made it possible to carry out such implementations at a higher level, making it simple to design even higher order filters. The use of boards like FPGAs, whose capacity to handle such operations is extremely efficient, is made necessary by the HDL implementation. Discussions around the MATLAB and Verilog programming language interface for the necessary analyses have begun. Also addressed in this work are several use cases where this interface may be excluded. These days, there is a tremendous demand for computerised models for all types of implementations. These might include the need for them in bio-medical applications, such as the need to identify and eliminate noise from ECG and EEG signals. These filters are also used for heave measurement in submarines. Multi rate filters combine these types of filters to change the frequency of the signals. In essence, the filters are used in all of these.

2. Methodology

The use of MATLAB to implement FIR filters offers a variety of possible approaches. One of these is to use the FDA tool, in which the interactive designer used to obtain the coefficients and the appropriate design code. The workspace receives the exported coefficients. Along with the provision of specific random input, the MATLAB code created by the tool was examined for its magnitude and phase responses in order to determine the properties of the filtered response. The general sinusoidal signal with additional noise was chosen as the random input.

The alternative method involved manually creating a

*Corresponding author: mahidharlolla1@gmail.com

MATLAB FIR filter code using window techniques. Kaiser Window was selected to be used out of a variety of commonly available windowing techniques. The major lobe width of the Kaiser window can be altered using a parameter. As a result, the filter response can be adjusted as needed. The data, however, demonstrate the implementation specifics. The findings are achieved by using the built-in functionalities of the filter once more to create the FIR filter using the Kaiser window. There is a contradiction in this as well. Although the Kaiser window has many benefits, its implementation on platforms for signal analysis like MATLAB is much more feasible. However, when it comes to tools based on hardware and real-time implementation, such as those using HDL, implementation becomes challenging. Therefore, in the subsequent portion of the work, it is suggested that changes to the architecture when done using HDL. The architecture has been optimised as a result of these changes. The above-mentioned process was executed and the results were analysed in detail.

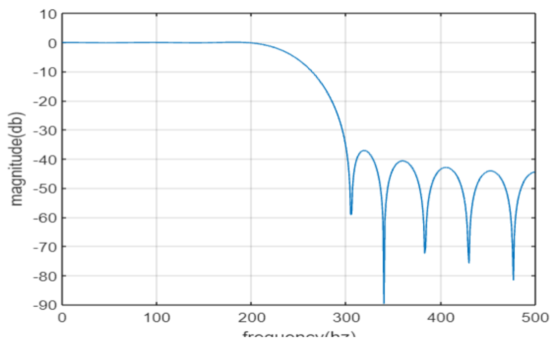


Fig. 1. Kaiser window magnitude response

There were several outcomes. The creation of the FIR filter using MATLAB code and straightforward built-in functions produced magnitude and phase graphs, which were then displayed using the plot function. Additionally, when a random input was presented, different responses were plotted and identified.

A. *Magnitude response vs. Output Response*

The filtered output stream is sometimes mistaken for the filter's overall magnitude response. In terms of waveforms, MATLAB implementation will show the differences between the two.

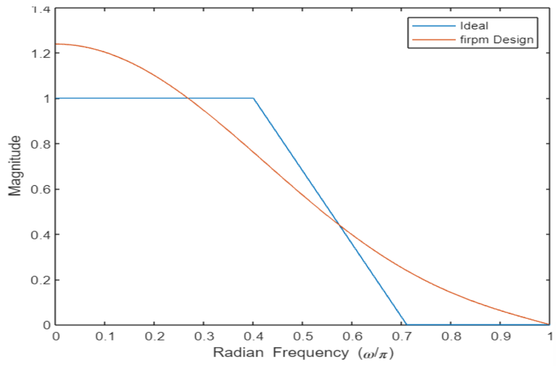


Fig. 2. Basic FIR magnitude and phase response

The charts for both replies were implemented using the "Plot" and "freqz" functions. If a plot function is supplied, the filter functions created using MATLAB code and built-in functions like firpm result in a plot of the magnitude responses as well as the phase response. Additionally, the plot function was now useful to depict the output filtered response when some random signal with noise like noise added sinusoidal signal is given as the input to the filter. Similar HDL implementation techniques make it possible to demonstrate that something is working by changing the amplitude of the input stream signal as shown by waveforms plotted against time. The most important thing to remember from this was that the output wave of the filter is less than the input wave at different stages (since LPF was taken into account).

B. *Group Delay*

For modulation, filters are frequently employed to reduce the signal's amplitude at various frequencies. Additionally, filters can cause variations in the phase of various frequencies, even when the amplitude is unmodulated. The timing information between different frequencies within the same signal and between separate signals may be disrupted by these phase shifts, which may result in time lags in the filtered signals. In the case of linear phase filters, such as FIR Filters, these time delays are constant, while in the case of non-linear phase filters, they change as a function of frequency. As a result, before sampling, timing information in signals may be distorted. These delays can be avoided in a variety of ways. This delay is seen not just in MATLAB but also in HDL-based implementation. There are a few potential solutions to this problem when taking into account the MATLAB implementation. Particularly, group delay has always been of interest in the FIR filters. Zero phase filters may be used to fix the phase delay. Zero phase shift is introduced for all frequencies. Research claims that these zero phase filters, however, cannot be used in online applications.

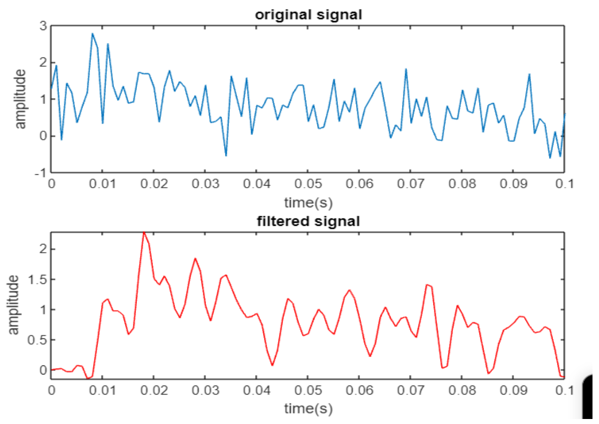


Fig. 3. I/O Response with group delay (180-degree phase shift)

Therefore, taking into account the online application, the original timing of the signal can be recovered by simply imitating the zero-phase filter's functionality once the parameters of the appropriate filter are properly determined, as is the case of MATLAB. As the coefficients of the zero-phase filter must be symmetric around zero, the real-time implementation of this zero-phase filter is currently not feasible. Inferentially, this suggests that the filter is not causal and that the current output depends on the input received in the future. Therefore, in this work, a method that may replicate this characteristic was sought after. Since the building of such a filter in real time is undoubtedly impossible, the use of MATLAB's "filtfilt" function is suggested here. The "filtfilt" function can simulate zero phase filtering during post-processing of the signal. This is a MATLAB built-in function. The following ideology for the mimicking process was discovered through implementation and analysis:

Since FIR filters have been taken into consideration, linear phase filters may not always cause phase distortion. It simply signifies that the output has been given a time shift. A constant shift with respect to time derives from a linear phase shift with respect to frequency. Simply expressed, the output will follow the input by a fixed number of samples. The group delay was specially formed by this.

If it is wished to keep the output signal aligned with the input signal, a track of the group delay must be kept. Simply expressed, the delayed version of zero-phase output is what is meant by linear phase output. As seen in the image below, the use of “filtfilt” to construct the filter has now made it possible to eliminate this delay instead of utilizing the standard filter functionalities.

In contrast, when HDL is used for implementation, what is referred to as offline filtering, a filter that functions similarly to an online filtering process is applied to the reversed signal, causing the phase shift to return to zero and restoring the timing of the filtered signal. This problem has been solved in the literature, and the design equations are used right here:

It is possible to build a pair of cascaded required filters (low pass is taken into consideration here) and all pass filters (which produce phase lead) so that the phase lag brought on by the required filters is counterbalanced by the phase lead generated by the all-pass filters. During the entire process of getting rid of the delay, it was discovered that when the impulse response of the forward and backward filtering was added up under the Non-Linear Phase (NLP) filtering process, the added impulse response was windowed to make it finite, and then the finite length kernel was delayed so that no additional samples were needed. Now, this can be used directly as the kernel for a FIR filter. The need for the interface between tools used to make filter designs may be eliminated by this technique.

C. Weighted Average Filter

Most individuals recognize filters in terms of frequency. The filter also has a time-domain explanation, as may be seen below:

Any signal, let's take an audio signal as an example, is a digital data made up of a number of digital samples. Simply put, each sample can be considered to be nothing more than a number. Therefore, it can be assumed—but this is simply an assumption and not actual fact—that an audio file is a collection of numbers stored in an array. Let's imagine that the first coefficient in the FIR filter is an amplification coefficient and the second coefficient is a multiplication coefficient for the previous sample. The amplification factor for the second previous sample is represented by the third coefficient.

Each of these coefficients is equivalent to multiplying a large number of serial data points by a variety of values. Let's say it is desired to remove all the high frequency components or spikes using a low pass filter. It is conceptually comparable to "averaging." All input samples are averaged in weighted fashion in FIR filters. Weights indicate the relative importance of each data point in advance. This indicates that since weight values cannot be negative, they must all be positive.

The averaging works best when the weights are similar, and vice versa. Lower cut off frequency here results from better averaging. When all the coefficients are identical, the weighted average becomes the simple average. The equalization of the frequency values of a data set is carried out via the weighted average.

The following is the FIR filter equation:

$$Y(n) = \Sigma\, b_k * x(n-k)$$
$$Y(n) = b_0.x(n) + b_1.x(n) + \cdots$$

The averaging is now understood to be the simple sum of the weighted function as in the equation above where the weighted functions b0, b1, b2, etc. Let's use the example of data samples a and b with weighted functions of 0.5*a and 0.5*b. A value equal to the average of these values is obtained when these values are added together. Here, samples in the order-of-time domain are being used. We need to determine the Discrete Time Fourier Transform (DTFT) and Discrete Fourier Transform (DFT) in order to determine the frequency domain of the collection of these samples. The reduction of the amplitude in the final response making weighted average a success.

D. Filter Analogies

An impulsive response that is smeared and needs to pass through the filter is considered. The filter's result is an impulse response that has been cleaned up. DSP blocks are now mostly utilized to quickly complete complex and specialized calculations. DSP processors, FPGAs, and ASICs can all be directly used to do these calculations. At this point, the digital filters can be viewed as a straightforward mathematical model. An equivalent mechanical mechanism is used here to specify the digital filter system:

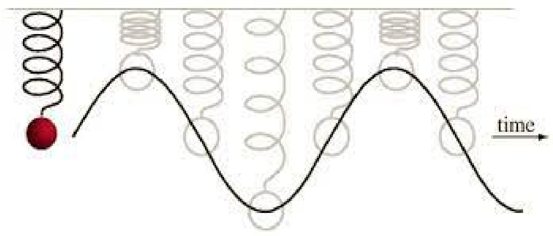


Fig. 4. Oscillations of a ball on a spring

Suppose a ball on the spring. This ball may roll up and down with certain oscillations and a gradually decreasing amplitude when an impulse input is applied to it.

If this is considered to be an IIR filter, then the resonance can forever take to delay. This is a different case with respect to the FIR filter since impulse response is finite. Upon further simplification, the equation is found to reduce to the form as follows:

$$y_o = 2y_1 - y_2 + y_1 k_c t - \frac{y_1 - y_2}{2m} * t^3$$

y_1 and y_2 are two consecutive positions of ball.
k_c is spring constant.
t is time and m indicates the mass of the ball.
y_o is the next position of the ball that is analogous to the next coefficient calculation in FIR filter.

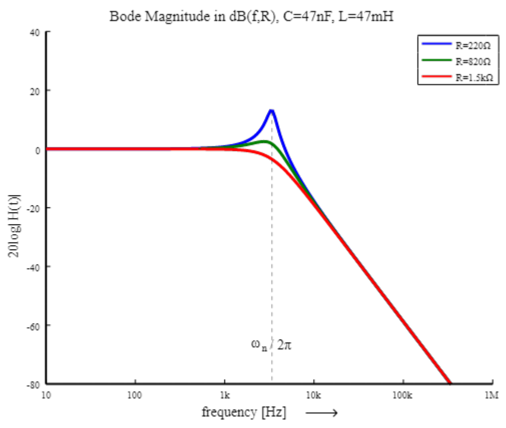


Fig. 5. Underdamped LPF response

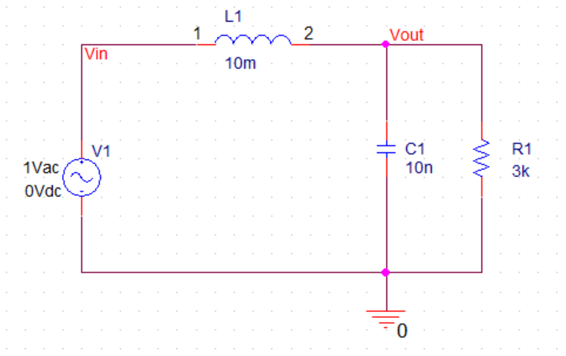


Fig. 6. IIR filter equivalent circuit

The above equation finally reduces to the form of average equations as shown below:

$$y_0 = a.y_1 + b.y_2$$

IIR filters model the actual physical process, whereas FIR filters just model the end result.

Take into account the filter's impulse response, as displayed below. This is taken from the MATLAB tool when FIR Filter is designed. It is found from the work that the filter coefficients are acquired for the FIR Filter when the impulse response of the filter is reversed. These coefficients are similar to the one's obtained from the FDA tool of MATLAB. As is customary, the input value is now multiplied by the matching samples to calculate the next sample.

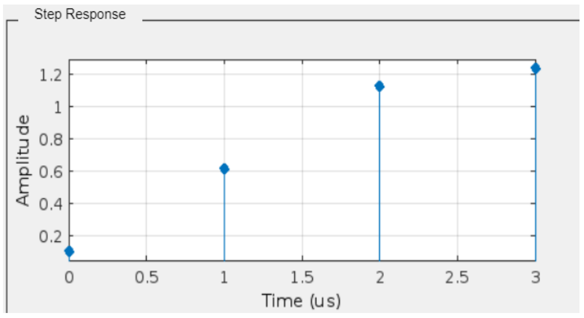
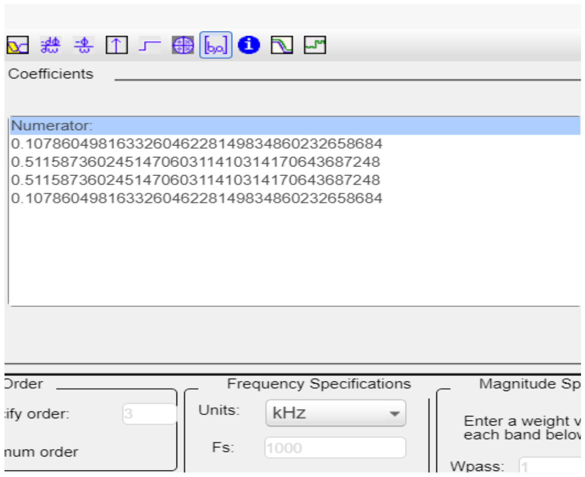


Fig. 7. Impulse response

Although it has been discovered that one of the filter arrays was actually flipped, this step is typically skipped because FIR Filter coefficients are largely symmetrical.



Coefficients

```
Numerator:
0.107860498163326046228149834860232658684
0.511587360245147060311410314170643687248
0.511587360245147060311410314170643687248
0.107860498163326046228149834860232658684
```

Order
ify order: 3
num order

Frequency Specifications
Units: kHz
Fs: 1000

Magnitude Sp
Enter a weight v
each band belov
Wpass: 1

Fig. 8. Coefficients (MATLAB)

As is already known, the impulse response of the FIR filter is truncated after the finite section.

There are two possible explanations for how the FIR Filter works.

- Each output sample is a weighted average of the most recent input sample, which is the first factor.
- With regard to the second, output is the culmination of all impulse reactions to every prior input.

Analogy 2:

Another example used is a loudspeaker, which is a frequent application for FIR filters. The loudspeaker output can be filtered by multiplying the filter coefficients using the speaker's output samples to obtain the summed weighted averages. It was intended to happen this way.

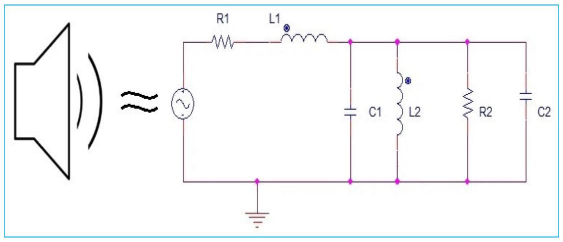


Fig. 9. Loud speaker equivalent circuit

However, there was no practical way to link the output of a loud speaker to the FIR Filter when the circuit tended to produce a mechanical output.

In order to make these effective loud speakers, commutative property can be used as a simple solution. The FIR Filter was supposed to be placed between the signal source and amplifier in this scenario. The audio signal can be passed as the input stream to the FIR filter coefficients multiplication stage. This signal is then given to the amplifier and thus, finally, amplifier output is taken. The output of the amplifier has a time smear. As a result, it is discovered that the FIR Filter stops the loudspeaker's resonance. The same type of idea can be applied to temporal equalization. The output of loudspeaker is an impulse as can be observed below:

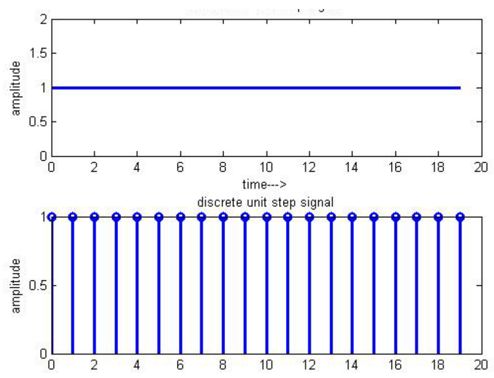


Fig. 10. Impulse to be provided in loudspeaker analogy

Since they only rely on the convolution between the kernel and the input, FIR filters are simple to create. Here, "Kernel" refers to the collection of filter coefficients. Convolution operations include multiplying the corresponding elements of two arrays of the same size and adding the results to get one output value. Convolution is a fundamental principle of FIR filters, but because it performs differently on different types of hardware, implementations including sequential, parallel, pipelined, moving average, and weighted average have been introduced. The input length is typically more than the kernel length in this case, but it can also be infinite, and the number of coefficients produced is the same as the number of taps that is further same as the number of the inputs.

E. Direct description of convolution

The previously described convolution process is directly described by the direct form structure, as seen in the image below. Shift registers might act like the memory's constituent parts. Here, simpler mathematical procedures are performed.

The drawback in this case is that depending on the sequence of the filter, more than two addition operations are involved in a single summation step. Although possible in HDL, this implementation is challenging. A too-long logical path between the input and the output can cause problems during implementation.

$$x(n) * h(n) = \sum_{k=-\infty}^{k=+\infty} h[k] * x[n-k]$$

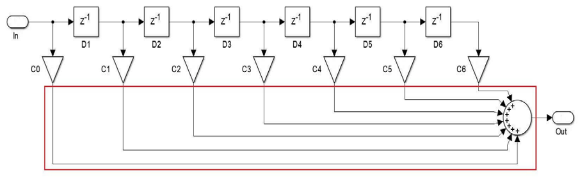


Fig. 11. Basic FIR

The convolution in the time domain is identical to the multiplication in the frequency domain, as is known from theoretical and mathematical models. This procedure was also impractical for implementation since frequency components can only be formed after they are amplified or attenuated, not when they are generated. The coefficients, not the structure itself, control the spectral properties. The pipelined form of implementation was chosen as an alternative to eliminate these concerns.

Pipelining involves shortening the distance between registers without affecting the device's functionality by changing the design or placing registers in between processes. This is advantageous for implementation in MATLAB and HDL. The addition is suited for hardware implementation because it is divided into several stages and separated by registers.

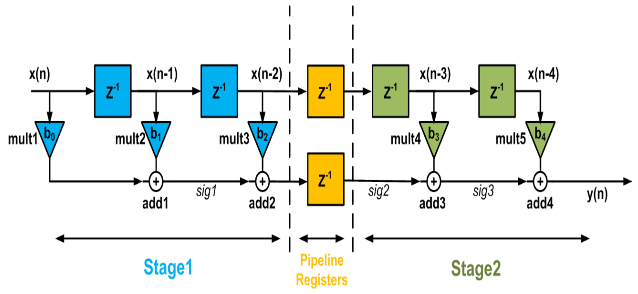


Fig. 12. Pipelined FIR

The architecture that is illustrated below makes input multiplication easier at every stage. As in the earlier systems, the direct input value is multiplied by the coefficients. Additionally, employing for loops is the most elegant approach to create such a filter. However, because these loops cannot be synthesized in languages like HDL, they are not advantageous for the real-time implementation of the filters.

3. Results

The various stages of FIR Filter implementation over the MATLAB tool provide the users with multitude of flexibilities. The common misconception of the magnitude response to the input/output response is cleared in the work discussed here so as to clearly identify and comprehend the meaning of magnitude response as well as the input, output response.

Another major issue behind the usage of the FIR Filter has been the potential group delay introduced in the final filtered response. This has also successfully been addressed and the comparison of both the responses is shown so as to indicate the elimination of the group delay.

Along with these details, the reduction in the magnitude of the output response indicating the effective operation of the low pass FIR filter has successfully been established. The spikes of the input signal higher in their amplitude are reduced to their specified minimal levels.

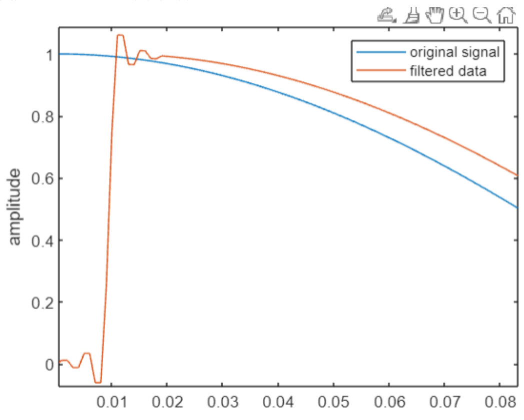


Fig. 13. I/O response

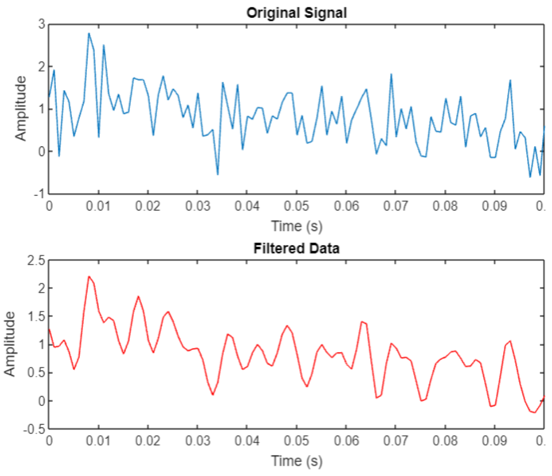


Fig. 14. Elimination of group delay indicated by zero phase change

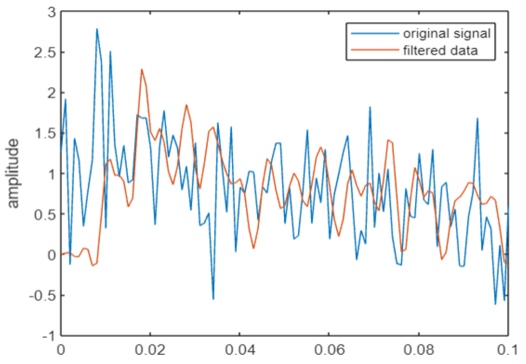


Fig. 15. Filtered signal with reduced amplitude

4. Conclusion and Future Work

The basic implementation and the case teachings have clearly been indicated. FIR Filters have their effective nature indicated when they are used in conjunction with their window techniques.

Although Kaiser window was used to implement the filter in the present work, the detailed analysis of the same is not provided. Hence, as part of the future work to be taken up, the functioning of FIR Filter with various windows like rectangular, bartlett, Hamming and Hanning window etc., may be considered.

Since, MATLAB is best proved to implement the FIR filters, the implementation has potential advantages when achieved on the reconfigurable logic and hence FPGA implementation for the window techniques-based FIR Filters may also be considered.

References

[1] S. Kumar, R. Mehra and Chandni, "Implementation and designing of FIR filters using kaiser window for de-noising of electrocardiogram signals on FPGA," 2016 IEEE 7th Power India International Conference (PIICON), 2016, pp. 1-6.

[2] R. Das, A. Guha and A. Bhattacharya, "FPGA based higher order FIR filter using XILINX system generator," 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), 2016, pp. 111-115.

[3] T. C. Singh and M. Kumar, "Digital FIR Filter Designs," 2021 Asian Conference on Innovation in Technology (ASIANCON), 2021, pp. 1-5.

[4] Niyama, Aaquib & Ramane, Sahil & Chhadwa, Neil. (2022). Implementation of IIR and FIR filters in Simulink MATLAB and its application in ECG.

[5] Rajesh Kumar Dwivedi, Raghav Dwivedi, 2017, FIR Filter Implementation using Matlab Fdatool and Xilinx Vivado, International Journal of Engineering Research & Technology, vol. 6, no. 10, October 2017,

[6] V. Dhillon, S. Nair, A. Pabarekar, M. Kumbhare, K. Thakur and R. Krishnan, "Implementation of FIR Digital Filter on FPGA," 2021 4th Biennial International Conference on Nascent Technologies in Engineering (ICNTE), 2021, pp. 1-5.