Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı 29, S. 273-277, Aralık 2021
© Telif hakkı EJOSAT'a aittir
Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue 29, pp. 273-277, December 2021
Copyright © 2021 EJOSAT
Research Article

FPGA Design of a High- Resolution FIR Band-Pass Filter by Using LabVIEW Environment

Güner Tatar [1*], Salih Bayar[2], İhsan Çiçek[3]

[1*] Fatih Sultan Mehmet Vakıf University, Faculty of Engineering, Departmant of Electrical-Electronic Engineering, İstanbul, Turkey, (ORCID: 0000-0002-3664-1366), gtatar@fsm.edu.tr

[2] Marmara University, Faculty of Engineering, Departmant of Electrical-Electronic Engineering, İstanbul, Turkey, (ORCID: 0000-0002-4600-1880), salih.bayar@marmara.edu.tr

[3] İstinye University, Faculty of Engineering, Departmant of Electrical-Electronic Engineering, İstanbul, Turkey, (ORCID: 0000-0002-7881-1263), ihsan.cicek@istinye.edu.tr

(International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT) 2021 – 21-23 October 2021)

(**DOI:** 10.31590/ejosat.1016363)

ATIF/REFERENCE: Tatar, G., Bayar, S., & Cicek, I. (2021). FPGA Design of a High- Resolution FIR Band-Pass Filter by Using LabVIEW Environment. *European Journal of Science and Technology*, (29), 273-277.

Abstract

Designers regularly use Finite Impulse Response (FIR) filters to fulfil the need for current electronic design applications such as signal or image processing and digital communications because of the remarkable selectivity computational efficiency. Fast and efficient information processing requires a dedicated microprocessor or a digital signal processor that may not always be available or provide enough performance. In such scenarios, designers can configure FPGAs for processing digitized signals. One of the most popular signal processing applications is filtering. Unlike the Infinite Impulse Response (IIR) filters, FIR filters do not have analog equivalent circuits. For this purpose, continuous time-discrete time conversion is not possible with the help of transforms. Because analog filters cannot have a finite impulse response, the design methods of FIR filters can be made as windowing method, pulse response truncation, and optimal filter design method. Considering this information, it aims to digitally separate two signals with different frequencies (2.4 kHz and 4.2 kHz), which are given to the input as analog, to obtain the desired information signal and suppress other signals. We preferred to use LabVIEW graphical programming language to get the digital FIR filter coefficients. We selected rectangular windowing, set the digital filter's sampling frequency as 18720 Hz, and determined the filter's coefficient with high-frequency resolution as 24. Using filter coefficients in the real-time FPGA-VHDL environment, we showed the performance and resource consumption. LabVIEW is used for simulation as well as obtaining filter coefficients. In addition, we compared both simulation and real-time FPGA-VHDL application output waveforms and examined both platforms' advantages and disadvantages.

Keywords: Digital filter design, Finite Impulse Response (FIR), FPGA-VHDL, LabVIEW Environment

LabVIEW Ortamını Kullanarak Yüksek Frekans Çözünürlüklü FIR Bant Geçiren Filtrenin FPGA Tasarımı

Öz

Tasarımcılar, dikkate değer hesaplama verimliliği nedeniyle sinyal işleme görüntü işleme ve sayısal iletişim gibi mevut elektronik tasarım uygulamalarına olan ihtiyacı karşılamak için düzenli olarak sonlu dürtü yanıtına (FIR) sahip filtreleri kullanılırlar. Verilerin hızlı ve verimli bir şekilde işlenmesi için her zaman mevcut olmayan veya yeterli performans sağlamayan özel bir mikroişlemci veya sayısal sinyal işlemcisi gerektirir. Bu tür senaryolarda tasarımcılar, sayısala çevrilmiş sinyalleri işlemek için FPGA'lerin yeniden yapılandırılabilir özelliğinden yararlanırlar. En popüler sinyal işleme uygulamalarından biri filtrelemedir. Sonsuz darbe yanıtlı (IIR) filtrelerin aksine, FIR filtrelerin analog eşdeğerleri yoktur. Bu amaçla dönüşümler yardımıyla sürekli – zaman, ayrık – zaman dönüşümü mümkün değildir. Analog filtreler sonlu bir darbe yanıtına sahip olmadığından, FIR filtrelerin tasarım yöntemleri; pencereleme yöntemi, darbe yanıtı kesme ve optimal filtre tasarım yöntemi olarak yapılabilir. Bu bilgiler ışığında girişe analog olarak verilen farklı frekanstaki (2.4 Khz ve 4.2 Khz) iki sinyali sayısal olarak ayırarak istenilen bilgi sinyalinin elde etmeyi ve diğer sinyalleri bastırmayı amaçlayan bir filtre tasarımı sunulmuştur. Sayısal filtrenin katsayılarını elde etmek için LabVIEW grafiksel programlama dilini kullanmayı tercih ettik. Tasarladığımız filtrenin yüksek frekans çözünürlüğü ile katsayısını 24, örnekleme frekansını 18720Hz ve filtreleme işlemi için dikdörtgen pencereleme yöntemini kullandık. Gerçek-zamanlı FPGA-VHDL ortamında belirlediğimiz filtre katsayılarını kullanarak performans ve kaynak tüketimini gösterdik. Bunlara ek olarak, hem benzetim hem de gerçek zamanlı FGPA-VHDL uygulaması çıkış dalga formlarını karşılaştırarak platformların avantaj ve dezavantajlarını inceledik.

Anahtar Kelimeler: Sayısal filtre tasarımı, Sonlu dürtü yanıtı (SDY), FPGA-VHDL, LabVIEW grafiksel tasarım platformu.

[*] Corresponding Author: gtatar@fsm.edu.tr

1. Introduction

Digital signal processing is widely used in many platforms such as multimedia and communications. In most of these applications, unwanted noise is generated due to the electrical components, bad circuit design, or the A/D conversion artefacts. The undesired noise signals must be eliminated for the proper operation of the system of interest. Conventionally, both analog and digital filters and filtering techniques have been used to suppress the noise and process the signal of interest in a selective manner. Discrete-time and continuous-time filter design options are also available according to the requirements of the target application. Thanks to the technological progress of A/D converters, modern signal processing is mostly performed in the digital domain. It is more convenient to use the digital environment because of the advantages and performance. Current semiconductor technology is focused on the development of higher speed digital circuits at submicron processes. In the digital domain, both hardware and software acceleration approaches are used to improve the performance of designs through parallelization. Hardware accelerators are used as system peripherals to offload the processor from computation-intensive tasks. Examples of hardware accelerators include multi-core CPU architectures, GPUs, application-specific integrated circuit ASICs, and FPGAs. Considering the cost, performance and energy consumption, the use of FPGAs in real-time applications is increasing [1-2]. FPGAs have been successfully employed in signal processing applications where both design flexibility and high performance are required. In digital signal processing, FPGA implemented digital filters are used frequently to remove interfering signals or noise.

In general, analog-to-digital converters are used to quantize analog expressions in a signal processing cycle digitally. The sampled and quantized information packet is then processed using the FPGA, as shown in Figure 1. The unwanted signals in the information signal are suppressed, and the processed signal is converted into a continuous signal with the help of a digital-analog converter. Signal processing in the digital domain allows for advanced algorithm designs that provide higher performance and greater flexibility than its analog counterpart. Developing semiconductor technology and increasing performance requirements in specific applications have made processing speed a critical parameter. And it made processing speed a required parameter. FPGA-based designs meet these requirements with low latency responses, high-performance workload, and parallel processing capability [3-4].

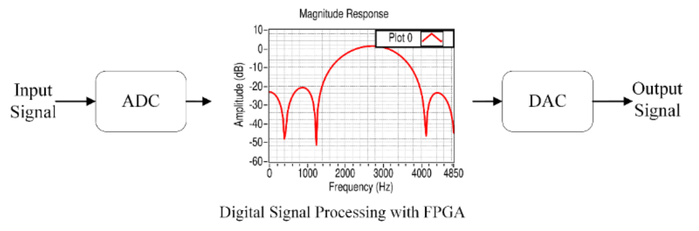


Figure 1. Block diagram of the designed band-pass FIR filter

In signal processing applications, filters perform the suppression of unwanted signals (noise). As mentioned above, filters can be discrete-time or continuous-time. We discussed discrete-time filter design here. A discrete-time filter consists of FIR and IIR [5]. Despite their low computational speed and high filter order, FIR filters are often preferred due to their linear phase and stability [6]. Designers can eliminate high computational load with the help of hardware accelerators with high switching frequencies such as FPGA. Since FIR filters do not have analog equivalents like IIR filters, they require direct discrete-time design. FIR filters, which are frequently used in applications where memory is not needed, and linear phase response is needed, have a stable output response for any input. This is the reason why FIR filters are preferred. With digital filters, the hardware design process can be a bottleneck for a successful outcome. Alternative high-level design methods such as National Instrumentation and Matlab have been developed to shorten the design time of the hardware. Although the traditional hardware description languages VHDL or Verilog are still used today, the need for high-level synthesis approaches has increased to shorten the design process.

This study examines high-frequency resolution FIR bandpass filter design as a sample design scenario to compare a digital filter design flow based on traditional hardware description language VHDL with a high-level design approach such as LabVIEW. The system diagram of the designed filter is given in Figure 1. The filter is designed with the Kaiser windowing method with 18750 Hz sampling frequency (FS), 1.8 kHz low cutoff frequency (FL), 3.6 kHz high cutoff frequency (FH). The designed filter suppresses high-frequency signals while allowing the low-frequency signal to pass through the sum of two different signals (2.4 kHz and 4.2 kHz) applied to the input, according to the determined cutoff frequencies.

We organised the rest of this study as follows; in Section II, we explained the high-frequency resolution bandpass FIR filter; in Section III, we discussed in detail the methods of obtaining the coefficients of the designed filter and the simulation results performed in the LabVIEW environment.

2. Design of the 24-Tap High Frequency Resolution FIR Band-Pass Filter

FIR filters are linear phase filters with a finite impulse response and have no analog equivalent. As the degree (tabs) of the FIR filter increases, the pass frequency range of the filter becomes narrower. While IIR filters consist of zeros and poles, FIR filters contain only zeros, so they are filters without feedback. Figure 2 shows the general block diagram of the FIR filter. Equations (1) and (2) are FIR filters' zero distribution and transfer functions.

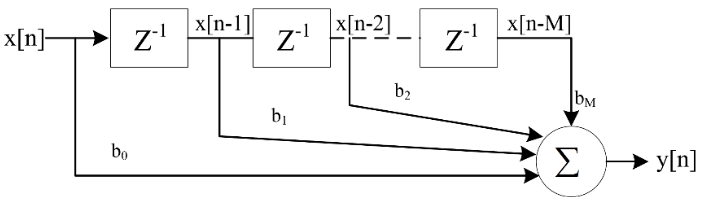


Figure 2. General block diagram of FIR filter

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] \tag{1}$$

$$H(z) = \sum_{k=0}^{M} b_k z^{-k} \tag{2}$$

We used Harris' rule-of-thumb method for filter design in terms of quick and convenient estimation. This basic rule provides the opportunity to obtain information about the number of tabs during filter design and perform it with the iteration method. Thus, we can see the effect of stopband attenuation and the steepness of the passband on the number of tabs. In equations (3) and (4), we found the value of tab 24 for a filter with 50 dB attenuation, a sampling frequency of 18750 Hz, a lower cutoff frequency of 1800 Hz and an upper cutoff frequency 3600 Hz.

$$B_T = \frac{F_{stop} - F_{pass}}{F_s} = \frac{3600 - 1800}{18750} = 0.096 \tag{3}$$

$$N_{Taps} = \frac{Attn(dB)}{22 \times B_T} = \frac{50}{22 \times 0.096} = 23.674 \cong 24 \tag{4}$$

Where, N_{taps} is simply called as a coefficients/delay pair, B_T is the normalized transition band, Attn is the desired attenuation in *dB*, F_{pass} and F_{stop} are the passband and stopband frequencies, F_s is the sampling frequency in Hz and 22 is constant value.

We carried out the design of the filter in the LabVIEW environment. We obtained the filter coefficients, and design results in the LabVIEW environment, then transferred them to the FPGA-VHDL environment and made comparisons accordingly.

2.1. Verification of the Discrete-Time FIR Filter and Calculation of the Filter Coefficients

We performed both simulation and verification of the filter using the NI LabVIEW platform. Here, we obtained the coefficients of the filter from the filter that we designed graphically. To get the coefficients, we used the sum of two signals, 2.4 Khz and 4.2 Khz, as the test signal and 18750 Hz. as the sampling signal. We preferred 90 degrees as the phase difference of the two signals with amplitudes of 0.8 V and 1.2 V. For the bandpass filter with 1.8 Khz lower cutoff frequency and 3.6 Khz uppercut frequency; we preferred 24th order and Kaiser window. Figure 3 shows the LabVIEW code block. As can be seen from the figure, the code block consists of three parts: the signal generation stage where we generated the test signal, the analog filtering and the digital filtering stage. Since there is no analog equivalent of FIR filters, there does not exist conversion between each other. Hence, we had to make a discrete-time and analog filter design. LabVIEW offers some programs as subprograms (sub. vi). Here, we used virtual instruments (vi.) provided by LabVIEW. The cutoff and sampling frequencies used in the discrete and continuous-time filtering stages are the same. We observed that the obtained filter coefficients are different. Since we designed the discrete-time filter, we used the coefficients of the digital filtering stage in the FPGA-VHDL environment, and the results were accordingly. In addition, amplitude response, phase response, Fast Fourier Transform (FFT), and discrete-time Z-plane show the filter's properties.

We calculated the frequency spectrum of the filtered signal using the Fast Fourier Transform (FFT) as shown in Figure 5-a. FFT is an algorithm where mathematical operations are done very quickly. While calculating the spectrum of signals, it uses the Discrete Fourier Transform (DFT), which is expressed by the equation (5) [7]. Computing the DFT for an application with N data requires approximately ~N^2 complex and time-consuming mathematical processes. For this reason, instead of directly applying DFT for an application with N data samples, we preferred to use the built-in FFT algorithm provided by LabVIEW and presented to the user out-of-the-box. Unlike IIR filters, since there is no feedback in FIR filters, there are only forward coefficients, which evolves to zero points. We obtained zero-point representations (5-b) of the filter using pole-zero.vi available in LabVIEW.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\left(\frac{2\pi nk}{N}\right)} \tag{5}$$

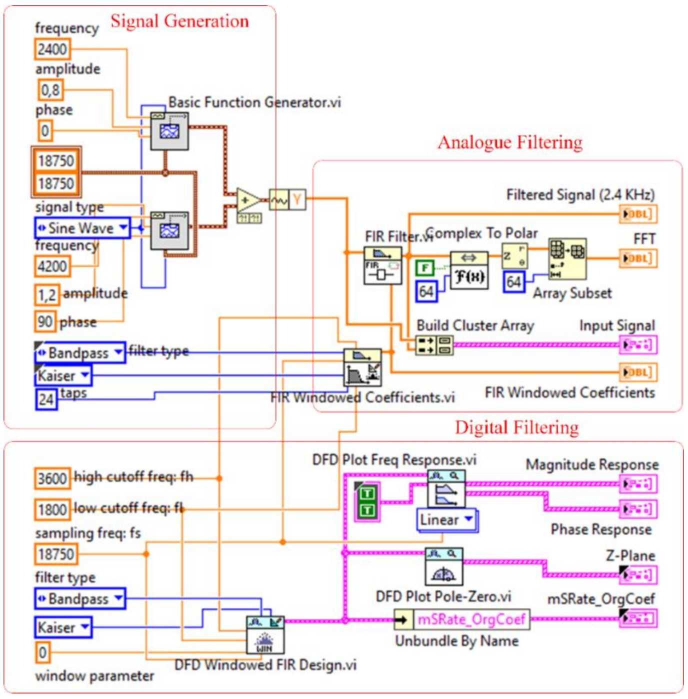
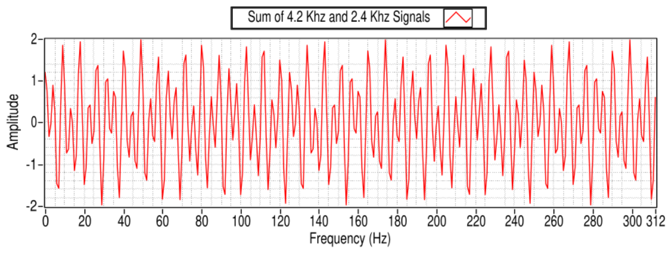
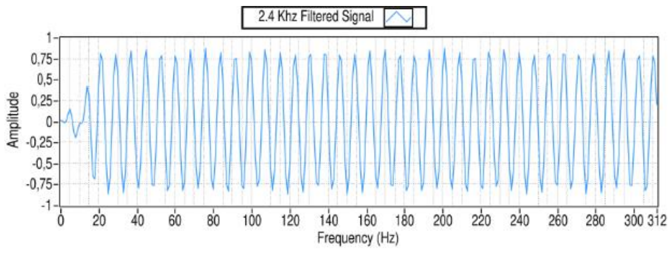


Figure 3. LabVIEW design of band-pass FIR filter with filter coefficients



a)



b)

Figure 4. Sum of 4.2 Khz and 2.4 Khz (a) signal output, and 2.4 Khz filtered (b) signal output

Figure 4-a gives the sum of the 4.2 Khz interference signal and the corresponding 2.4 Khz signal, whereas Figure 4-b shows the filtered signal. As seen in the filtered signal, we can observe that the output signal is about 0.8 V and 2.4 Khz. Since the signal of interest stays in the passband range, it allows the filter output, while the noise signal is stopped by the filter corner frequencies and is not allowed to pass through.

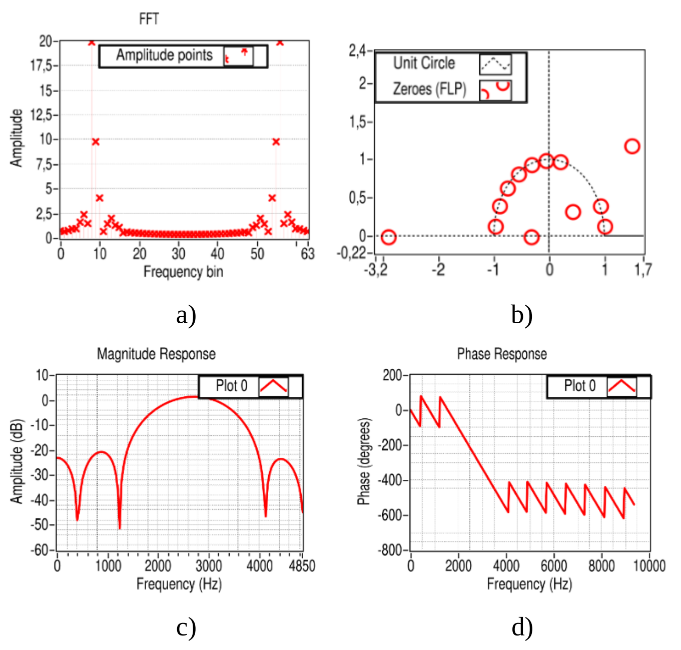
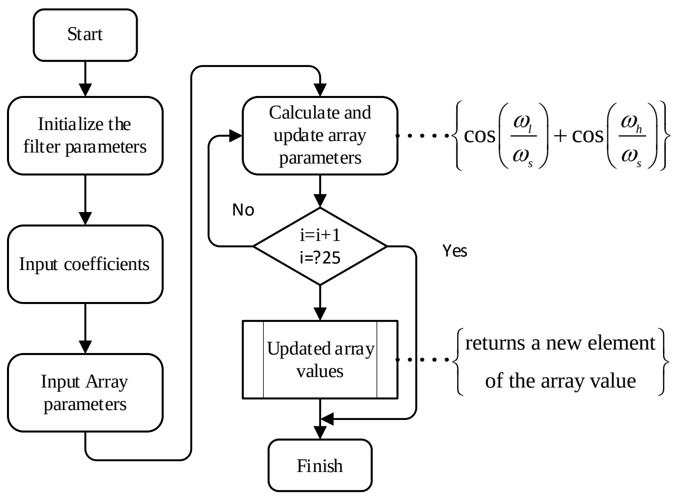


Figure 5. Fast fourier transform (a), zero location (b) on the complex z-plane, Magnitude (c) response, and phase (d) response of the FIR filter

One can find the magnitude response of an FIR filter by interpreting a linear system. The real magnitude spectrum is defined as the input to a linear system with the magnitude response of a rectangular window. The output obtained in response to this input is obtained as the magnitude response of the FIR filter. Considering the characteristics of linear systems, the output of a system appears as the product of the input and the system response. However, different window types affect the shape of the magnitude response. Figure 5-c shows the frequency spectrum of the magnitude response of the Kaiser filter. Suppose it is desired to design a filter with a better transition region approximation. In that case, it is recommended to choose window functions with different frequency characteristics in accordance with the magnitude response. Here, the Kaiser window model is preferred by considering the filter's structure, lower and upper cut-off frequencies and sampling frequency. Figure 5-d gives the phase response of the system.

3. VHDL Implementation and Algorithm Design

We used the flowchart given in Figure 6 and the pseudocode next to it for the VHDL implementation. VHDL, which is one of the hardware description languages, was preferred for the FPGA platform. The VHDL language consists of the *entity* layer where the input-output definition is made, an *architecture* that performs the logic functions, and a *process* layer that performs serial programming. We created an array of 24 in the architecture layer to hold the filter coefficients we obtained in the LabVIEW environment and two integer arrays to hold the input signal and filtered signals. Since the clock speeds of each hardware are different, we defined the source frequency after writing the code, not at the input. Considering the rising edge of the clock frequency in the process, we designed a shift register for each clock transition. When the program is started to run, filter coefficients are multiplied by each register cell according to the determined parameter values, shifted and saved. This process continues according to the value of the counter in the "*for* loop". A continuous-time signal is generated from the discrete-time, accordingly. Here, we preferred direct form FIR filter design and compared both LabVIEW and VHDL results.


Start
Initialize the filter parameters
Input coefficients
Input Array parameters
Calculate and update array parameters
$\left\{\cos\left(\frac{\omega_l}{\omega_s}\right)+\cos\left(\frac{\omega_h}{\omega_s}\right)\right\}$
No
i=i+1
i=?25
Yes
Updated array values
returns a new element of the array value
Finish

Algorithm 1 Discrete-time FIR Filter Design

```
1:  Initialize the filter parameters
2:  Define the clock frequency and Array size
3:  Compute filter coefficients
4:  for iteration = 1, 2….. do
5:          Update array parameters
6:          if the filteredSignal obtained
7:          then
8:          Generate output signal
9:  end for
10: Implement as an hardware
```

Figure 6. Flowchart and algorithm of FIR filter implementation

The algorithm above provides the VHDL implementation of the FIR filter. For the easy application of the algorithm, the filter's lower and upper cutoff frequencies and sampling frequency are determined. One should adjust the clock frequency according to different hardware features. Upon the program execution, we multiplied the input signal and the filter coefficients. Because the filter is in the pipeline structure, we kept the previous value of the filter output in memory in each clock cycle. This situation continues until all the coefficient values are updated. We calculated the filter order, lower cut and uppercut frequencies and sampling frequencies for the VHDL implementation. We tested them in the Aldec Active-HDL simulation environment and obtained the results presented in Figure 7.

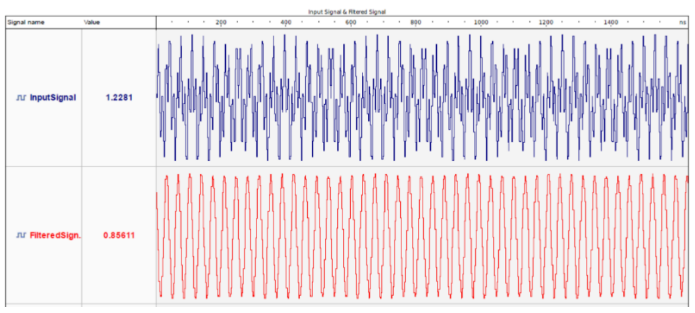


Figure 7. VHDL implementation functional simulation resul

As seen in Figure 7, we obtained results similar to LabVIEW design results. The result of the application shown in blue in Figure 7 shows the sum of the two signals applied to the input, and the filtered signal in red. It is essential to make the necessary optimizations on hardware with limited resources. Table 1. shows the estimated resource usage in the FPGA by LabVIEW of the work done. The resource usage in the given table is calculated as an estimate. The designer can reduce resource usage with different algorithms and optimization methods.

Table 1. LabVIEW approximate FPGA Resource Utilization

Data type	**FF**	**LUT**	**BRAM**	**MUX**	**DSP**
Function generator	94	156	2	-	-
FFT	697	2368	6	16	-
FIR filter vi	1000	1200	-	-	12
Other remaining usage	20473	21359	82	-	138
Total usage	22264	25083	90	16	150

4. Conclusions and Recommendations

In this study, we designed a high-frequency resolution (24 tabs) discrete-time direct form bandpass FIR filter both in the LabVIEW environment and in real-time in the FPGA environment. We obtained the filter coefficients for the FPGA-VHDL application from the LabVIEW design. According to the given corner frequencies (3.6 Khz upper, 1.8 Khz lower), the designed filter passes the low frequencies from the sum of two signals with different frequencies (2.4 Khz and 4.2 Khz) while detecting other signals as noise and suppressing them. We determined the sampling frequency of the filter to be 18750 Hz. and its order to be 24. We developed and implemented the VHDL model to validate the functionality of the proposed filter. The simulation and application results show that the filter works without any problems. Considering the LabVIEW-FPGA resource usage, we observed that the designed filter has a compact footprint, indicating a suitable basis for its integration.

On the one hand, we designed the filter in LabVIEW, in which designers can develop programs quickly without struggling with text-based languages. On the other hand, the VHDL environment is more suitable for hardware designers interested in hardware description languages and successful in software languages. It is not possible to synthesize a program without arithmetic optimization. In FPGA-based hardware, the numbers must be fixed or floating-point; this is tiring and time-consuming. If the designer is going to do HDL programming, he/she should be aware of them. Therefore, LabVIEW is more suitable for rapid prototyping. Transferring the code written in the LabVIEW environment to the FPGA environment is possible with the LabVIEW-FPGA ready software package. All the designer has to do is compile the written code to suitable hardware. Although the VHDL language is difficult to learn compared to other languages, its human readability ensures that it can interfere with the code at any point. Since LabVIEW is graphical programming, there is almost no chance of intervention for each piece of code.

References

[1] Özpolat, E., Karakaya, B., & Gülten, A. (2017). FIR Filtre Tasarımı ve FPGA Ortamında Gerçeklenmesi. Fırat Üniversitesi Mühendislik Bilimleri Dergisi, 29(2), 269-275.

[2] Singh, G., & Prakash, N. R. (2017). FPGA implementation of higher order FIR filter. International Journal of Electrical and Computer Engineering, 7(4), 1874.

[3] Paul, A., Khan, T. Z., Podder, P., Hasan, M. M., & Ahmed, T. (2015, February). Reconfigurable architecture design of FIR and IIR in FPGA. In 2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN) (pp. 958-963). IEEE.

[4] Tatar, G., Kılıç, O., & Bayar, S. (2019, November). FPGA Based Fault Distance Detection and Positioning of Underground Energy Cable by Using GSM/GPRS. In 2019 International Symposium on Advanced Electrical and Communication Technologies (ISAECT) (pp. 1-6). IEEE.

[5] Pal, R. (2017, December). Comparison of the design of FIR and IIR filters for a given specification and removal of phase distortion from IIR filters. In 2017 International Conference on Advances in Computing, Communication and Control (ICAC3) (pp. 1-3). IEEE.

[6] (2021, August 5). Know all About FIR Filters in Digital Signal Processing. https://www.elprocus.com/fir-filter-for-digital-signal-processing/.

[7] Tatar, G., Cicek, I., & Bayar, S. (2021). FPGA Design of a Fourth Order Elliptic Band-Pass Filter Using LabVIEW. European Journal of Science and Technology, (26), 122-127.