FPGA-based Implementation of IIR Filter for Real-Time Noise Reduction in Signal

Aladin Kapić[1], Rijad Sarić[1], Slobodan Lubura[1, 2], Dejan Jokić[1]

[1]Faculty of Engineering and Natural Sciences, International Burch University (IBU), Sarajevo, Bosnia and Herzegovina

[2]Faculty of Electrical Engineering, University of East Sarajevo (UES), East Sarajevo, Bosnia and Herzegovina

aladin.kapic@stu.ibu.edu.ba

rijad.saric@stu.ibu.edu.ba

dejan.jokic@ibu.edu.ba

slobodan.lubura@etf.ues.rs.ba

***Abstract*—Filtering of unwanted frequencies represents the main aspect of digital signal processing (DSP) in any modern communication system. The main role of the filter is to perform attenuation of certain frequencies and pass only frequencies of interest. In a DSP system, sampled or discrete-time signals are processed by digital filters using different mathematical operations. Digital filters are commonly categorized as Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). This research focuses on the full VHDL implementation of digital second-order lowpass IIR filter for reducing the noisy frequencies on the FPGA board. The initial step is to determine, from continuous time domain function, the transfer function in the complex {s} domain, then map transfer function in complex {z} domain and finally calculate the difference equation in discrete-time domain of the system with adequate coefficients. Prior to the FPGA implementation, the IIR filter is tested in MATLAB using a signal with mixed frequencies and signal with randomly generated noise. The digital implementation is completed by using fixed-point binary vectors and clocked processes.**

Keywords-digital signal processing; IIR filter; digital design; FPGA; VHDL; Bode diagram

1. Introduction

Every modern communication system or architecture has at least one digital signal processing (DSP) module. The most efficient method to clear out noisy signals represent the use of signal processing technology called filtering. A filter represents a circuit design to allow the passing of a particular frequency band as well as attenuation of the signal outside the considered band. In a general view, filters can be divided as analog and digital. However, digital filters have numerous advantages over analog which makes them the core component of almost every digital system nowadays. Some of the digital filter advantages are high accuracy, parameter stability, precision, reliability, and small circuit size. Indeed, they have better performance than conventional analog filters, in almost every situation, especially

in case of insufficient stability of analog filters caused by temperature, tolerance and aging. Digital filters can be used for DSP applications where the sharp cut-off edge is desired [1, 2].

In the past decades, the rapid development of medium-scale integration (MSI), large-scale integration (LSI), very-large-scale integration (VLSI) and extremely large-scale integration (VVLSI) circuits affect the application of digital filters. Hence, digital filters are currently a regular element of cell phones and PCs. They are utilized for audio or image signal (1D/2D) filtering, noise reduction, etc. In the Linear-Time Invariant (LTI) systems, input signals usually contain a mixture of desired and undesired frequency components. In such a case, the filter is necessary to remove any type of noise presented in the signal [3]. Filters are mainly designed in the frequency (s or jω) domain by finding the most acceptable frequency response. The common issue is determining a practically realizable filter that has an adequate approximation of magnitude and phase. Filters can be classified as Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). The main characteristic of FIR filters is a finite period or limited duration of the impulse response. This filter contains only zeros and either no poles or poles only at the origin [4]. Contrary to that, the IIR filter (recursive filter) has an unlimited duration of the impulse response. The transfer function G(s) of IIR filter is calculated in continuous-time (CT) and the design of the filter is realized by establishing a recursive structure. It always contains a feedback element and both poles and zeros in the stability diagram. Preferably, it is mapped from {s} to {z} domain applying some of the CT to DT (discrete-time) transforms such as Bilinear transformation. The output from the IIR filter is distorted in phase, which does not happen in FIR filter [5].

According to [6] The rapid progress of VLSI circuit technology causes a high demand for high-speed and reprogrammable circuits for conducting digital design for real-time signal processing. Although the application of microcontrollers is presented in a considerable amount of digital systems, Field Programmable Gate Arrays (FPGAs) demonstrate superiority in terms of reconfigurability, high density of logic circuits, low-cost, and high reliability. For example, FPGA with the same base clock frequency as a microcontroller could handle almost twice the input signal frequency. Additionally, FPGA sampling rate is independent of filter order plus latency of microcontroller in the context of magnitude is much greater compared to FPGA.

A fundamental FPGA-based chip is defined as a matrix of configurable blocks (combinatorial or sequential circuits) linked using entirely reprogrammable interconnections where memory cells control the logic blocks [7]. Programming complex logic circuits for prototyping design including synthesis for verification purpose on FPGA is undertaken using Very Hardware Description language known as VHDL/Verilog. After verification of prototyping design on FPGA, further implementation on Application-Specific Integrated Circuit (ASIC) could be possible. Finally, FPGAs have found their applications in different industrial areas such as medical robotics, automated driving, aircraft embedded control and similar fields [8]. In recent years, FPGA-based embedded systems are used for the implementation of machine learning algorithms i.e. Artificial Neural Networks (ANNs) for processing EEG signals [9].

This paper aims to establish the digital design of the IIR filter on FPGA embedded platforms using direct VHDL coding. The initial phase of the design methodology starts by creating RC circuit which indicates analog lowpass filter. After that, the transfer function of the system is calculated, and filter design is analyzed in Laplace's or {s} domain. Also, it is necessary to map the obtained transfer function into the discrete domain and derive its parameters of difference equation. Finally, using the derived difference equation, the parameters of the designed filter are adjusted in MATLAB for further implementation on the FPGA.

2. Methods and materials

As stated in [10] IIR filters have infinitely long responses, which allows design in CT. The specifications of practical filters can be categorized as lowpass, bandpass and high pass, but a filter also can be designed by combining mentioned categories. The system function of causal, stable, and realizable IIR filter in DT can be represented in terms of impulse response and coefficients of difference equation given in (1)

$$H(z) = \sum_{n=0}^{\infty} h[n]z^{-1} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1+\sum_{k=1}^{N} a_k z^{-k}} \quad (1)$$

where $h[n] = 0$ for $n < 0$, and all poles are inside a unit circle ($z = |1|$) with zeros that can be located everywhere.

Procedure discussed in this section approximate the magnitude-squared response of ideal lowpass filter with cut-off frequency Ω_c, which is described in (2)

$$|H_d(j\Omega)|^2 = \{1, \quad 0 \le |\Omega| \le \Omega_c \; 0, \quad |\Omega| > \Omega_c \quad (2)$$

Practical lowpass filter with cut-off frequency (Ω_c) has a transition band between passband and stopband. Tolerance diagram of the analog lowpass filter is depicted in Fig. 1.

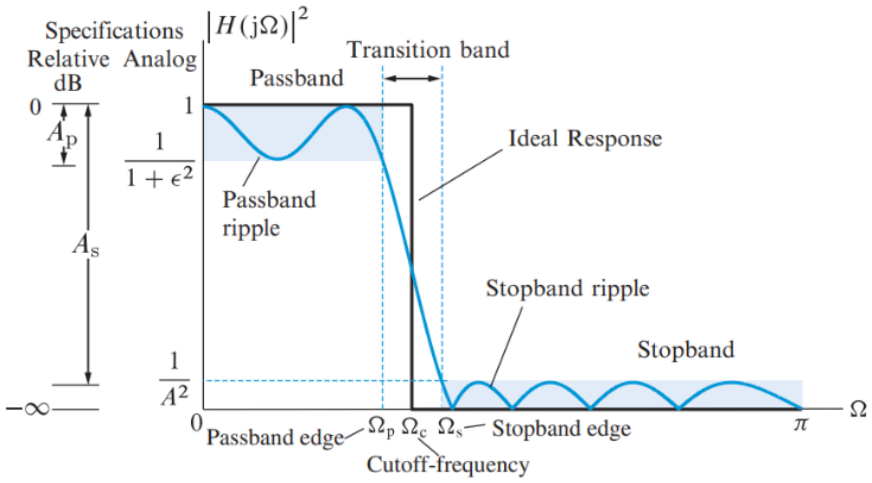


Figure 1. Tolerance diagram of low pass analog filter [10]

In Fig. 1, Ω_p and Ω_s are passband and stopband edges of the filter. Referring to (2), the approximation function must estimate $f = |H_d(j\Omega)|^2$, where $f \cong 1$ for $|\Omega| < \Omega_c$ and zero for all $|\Omega| > \Omega_c$.

Numerical description of classical approximation technique is represented in (3)

$$|H_d(j\Omega)|^2 = \frac{1}{1 + V^2(\Omega)} \tag{3}$$

where $V^2(\Omega) \ll 1$ for $|\Omega| \leq \Omega_c$ including $V^2(\Omega) \gg 1$ for all $|\Omega| \geq \Omega_c$ [10].

The ordinary, second-order, analog filter is created by connecting in series two first-order filters (one-pole filters), by making a serial connection of resistor and capacitor as shown in Fig. 2.

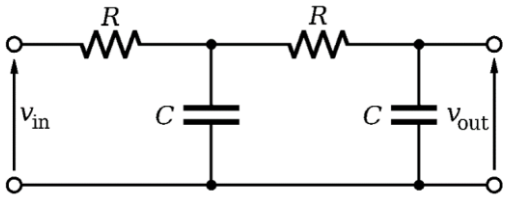


Figure 2. The standard schematic of the analog *RC* lowpass filter of the second-order

The reactance of a capacitor varies inversely with frequency, while the value of the resistor remains constant as the frequency changes. At low frequencies, the capacitive reactance ($X_c = \frac{1}{2\pi fC}$) of the capacitor is very large compared to the resistive value of the resistor (R). Indicating that the voltage potential across capacitor (V_c) is much larger than the voltage drop caused across the resistor (R). At the higher frequencies, the process is reversed. Let assume a first-order filter. The standard differential equation for first-order analog filter is described in (4)

$$u(t) = u_R(t) + u_C(t) = RC\frac{dx(t)}{dt} + x(t) \tag{4}$$

where $u_R(t) = i(t)R$, $i_c(t) = i_R(t) = C\ \frac{dU_c(t)}{dt}$. After applying Laplace transform on (4), the outcome is described in (5) and (6):

$$U(s) = RC\ sX(s) + X(s) = X(s)[1 + sRC] \tag{5}$$

$$G(s) = \frac{X(s)}{U(s)} = \frac{1}{1 + sRC} \tag{6}$$

G(s) is a transfer function of given system, X(s) and U(s) are input and output of system. Cut-off frequency for this lowpass filter is given as $f_c = \frac{1}{2\pi RC}$. Hence, combining values of resistor and capacitor, cut-off frequency can be shifted between 0 and π. The easiest way to increase order of filter is to bind more first-order filters, in a cascade structure. In that case, response become steeper for every block added to structure. In Laplace {s} domain, transfer function is changed with filter order, as in (7)

$$G(s) = \frac{X(s)}{U(s)} = \frac{1}{(1 + sRC)^n} \tag{7}$$

where n is the n-th order of the filter. Thus, the Laplace transform for the second-order filter is given by (8)

$$G(s) = \frac{X(s)}{U(s)} = \frac{1}{(1 + sRC)^2} \tag{8}$$

As it has already mentioned that, the output from IIR filters is distorted in phase, the most applied method to draw the frequency response of system is Bode plot. The Bode plot for LTI system, with a transfer function $G(s)$, consists of a magnitude and a phase plot. The Bode magnitude plot is the graph of the function $|G(s = j\omega)|$ of frequency ω with

j being the imaginary unit. The ω-axis of the magnitude plot is logarithmic and it is given in decibels (dB), i.e. a value for the magnitude $|G|$ is plotted on the axis at $20log_{10}|G|$. The Bode phase plot is the graph of the phase, commonly expressed in degrees, of the transfer function $arg\ arg\left(G(s=j\omega)\right)$ as a function of ω. The phase is plotted on the same logarithmic ω-axis as the magnitude plot, but the value for the phase is plotted on a linear vertical axis. Using the described method, phase distortion is displayed in a simplified form [11]. Calculation of the Bode plot from $G(s)$, where $s = j\omega$, is represented in (9) and (10)

$$G(j\omega) = \prod_{i=0}^{n-1} G_i(j\omega_i) = \prod_{i=0}^{n-1} |G_i(\omega)e^{j\varphi_i(\omega)} \tag{9}$$

$$log\ log\ [G(j\omega)] = \sum_{i=0}^{n-1} |G_i(j\omega)| + i(\sum_{i=0}^{n-1} \varphi_i(\omega))log(e)] \tag{10}$$

While increasing the filter order, a signal decrease of 20 dB per decade, which could be noticed in the plot. Thus, depending on the filter application, the order can be adjusted. In case of all filter frequencies above 50 Hz, with the decrease of 40 dB/decade (6 dB at cut-off frequency of 50 Hz), the second-order filter is designed, with values for R and C as 3.18 $k\Omega$ and $1\mu F$. In this situation, transfer function in $\{s\}$ domain is given as (11)

$$G(s) = \frac{X(s)}{U(s)} = \frac{1}{(1+ks)^2} \tag{11}$$

where k = $RC\ [\Omega F]$. Generating transfer function from complex $\{s\}$ domain to complex $\{z\}$ domain, Tustin's bilinear transform was used. This transformation is a first-order approximation of the natural logarithm function, that represents an exact mapping of the z-plane to the s-plane. In that case, $s = \frac{2}{T}\frac{z-1}{z+2}$, where T is sampling period. The final form of $G(z)$ is given in (12)

$$G(z) = G(s)|_{s=\frac{2z-1}{Tz+1}} = \frac{T^2(z^2+2z+1)}{(T^2+4kT+4k^2)z^2+(2T^2-8k^2)z+(T^2-4k^2)} \tag{12}$$

To normalize coefficient by z^2, the equation was divided with $(T^2+4kT+4k^2)$, which is noticeable in (13)

$$G(z) = \frac{\frac{T^2(z^2+2z+1)}{(T^2+4kT+4k^2)}}{z^2+\frac{(2T^2-8k^2)}{(T^2+4kT+4k^2)}z+\frac{(T^2-4k^2)}{(T^2+4kT+4k^2)}} \tag{13}$$

where $k = 0.00318$ and $T = 0.0001$, the final calculation of the transfer function of the system $G(z)$ is given as (14)

$$G(z) = \frac{Y(z)}{X(z)} = \frac{0.0002396z^2 + 0.0004793z + 0.0002396}{z^2 - 1.938z + 0.939} \tag{14}$$

Implementation of IIR filter on FPGA or microcontroller requires the transformation of obtaining transfer function into discrete difference equation, using reverse Z-transform which is calculated in (15) and (16)

$$Y(z)Q_m(z) = X(z)Q_r(z) \tag{15}$$

$$Y(z)(z^2 - 1.938z + 0.939) = X(z)(0.0002396z^2 + 0.0004793z + 0.0002396) \tag{16}$$

For the easier implementation of IIR filter, and taking lower number of bits to store data, coefficients were rounded to three decimals, so (16) is approximated to (17)

$$Y(z)(z^2 - 1.938z + 0.939) = X(z)(0.0002z^2 + 0.0004z + 0.0002) \tag{17}$$

To shift data in the discrete-time domain, (17) needs to be divided with z^2, and it yields (18)

$$Y(z)(1 - 1.938z^{-1} + 0.939z^{-2}) = X(z)(0.0002 + 0.0004z^{-1} + 0.0002z^{-2}) \tag{18}$$

In the discrete-time domain (18) is further simplified to obtain the final difference equation with coefficients of designed IIR filter as (19)

$$y[n] = y[n-1] * 1.938 - y[n-2] * 0.939 = \\ x[n] * 0.001 + x[n-1] * 0.002 + x[n-2] * 0.001 \tag{19}$$

The final form of the second-order IIR lowpass filter was designed for full manual implementation on FPGA using VHDL coding, Intel Quartus Prime 18.1 and ModelSim tool for simulation of the filtering process.

3. FPGA-Based design of the digital IIR filter

Prior to VHDL coding, testing the designed IIR filter in the MATLAB software environment is the initial step. It is more useful to generate and analyze the output of filtered signals in MATLAB and afterwards perform digital design

for implementation on the FPGA. In the first part, two sine wave signals of different frequencies were selected to examine the output of the designed filter in MATLAB. The first signal was represented as $x(t)=1/2\ \sin(2\pi*f*t)$, where f = 50 Hz or 50 sine wave cycles per second together with an amplitude (A) of 0.5. The sampling rate (Fs) of 1 MHz or 1 million samples per second was selected since the FPGA integrated 12-bit Analog-to-Digital Converter (ADC) (f_{ADC} = 16 MHz and 4096 resolution) depends on the base clock frequency of 50 MHz available in DE0-Nano FPGA development board with Cyclone IV chip. The max frequency that allows generating samples on FPGA is 1 MHz through the serial communication which sends 12 bits or bit-by-bit (50/16 MHz = 3.125 MHz). Following, the time array (t) was created by starting at 0 and incrementing by sampling period of $Ts = 1/Fs = 1$ μ sec (0.001 milliseconds). The sample generation ended when $t = NoC$ (number of sine cycles = 100) * $1/f$, which is two seconds. In other words, two seconds are needed to generate a sine wave with 100 cycles or 2 million samples with a predefined sampling rate. The resulting signal $x(t)$ contained 20 000 samples per each sine wave of $\omega = 2\pi\ rad = 314^o$ to produce exactly 50 Hz of base signal frequency. In general, the more samples generated per second (>>Fs) makes the sine wave smother and enhances the performance of the system.

(20)

Moreover, the second signal was represented as $x_1(t)=1\ \sin(2\pi*f_1*t)$, where f_1 is the frequency of 1 kHz for the same amount of time ($t = 2\ sec$) and A = 1. Again, the second signal contained the same number of samples (2 million), but 1000 cycles including 1000 samples generated in the single sine wave. These two signals $x(t)$ and $x_1(t)$ were added together to generate composite/mixed-signal $y_1(t)$ or out-of-bound signals with two distinct frequencies 50 Hz and 1 kHz. The new signal $y_1(t)$ has no longer **fixed** frequency, however, it is normally considered as the signal with baseband of 50 Hz as well as carrier frequency of 1 kHz. Owing to the low order of the IIR filter, the desired performance can be achieved with noise over 1 kHz. Lastly, the third signal was represented as $y_2(t)=0.2\ \sin(2\pi*f_2*t)$ + 0.2 randn(size(t)) where f_2=50 Hz, time (t=2 seconds) and A = 0.2. The third signal was composed of a sine wave of A = 0.2 and randomly generated white noise of the same aolitude with equal intensity at different frequencies. The A = 0.2 was selected to prevent the generation of samples out of defined ADC range.

In essence, the first composite signal $y_1(t)$ contains values between -1.5 and 1.5 and the second signal $y_2(t)$ with noise has values from 0.8 to -0.8. These floating-point numbers (if the angles on the unit circle are between 90^o – 0 – -90^o) cannot be directly implemented on FPGA. In fact, each number less than one cannot be represented on FPGA since electronic circuits can only represent values in standard 2^n format. Therefore, to make it suitable for FPGA representation the scaling of raw data is necessary by converting it into integer values ranging from 0 to 4095, where 0 represents input signal of $-V_i[V]$ while 4095 represents value of $+V_i[V]$. Original $y_1(t)$ and $y2(t)$ sine values were multiplied by 4096 to fit in a 12-bit binary number and then divided by 3.328. Division by numbers slightly larger than three was required since sample values of both signals can be negative, and one bit should be reserved for the sign of binary representation of integer numbers on FPGA. All samples can be as large as 1.5 or -1.5, so they need to be divided by at least 2 * 1.5 because of making some headroom for the sign bit. The finishing step stands for saving the integer portion of the result. The equation for calculation values suitable for FPGA description in look-up table (LUT) is given by 20

$$V_i(kT) = \frac{V_i(kT) * 4095}{3.328} \tag{20}$$

where $V_i(kT)$ represents the values from out-of-bound and noisy signals. The expressed difference equation (19) of IIR filter is utilized by iterating 20 000 times over every scaled sample and storing belated samples *x[n-1]* and *x[n-2]* (older values) together with older results of calculation of difference equation *y[n-1]* and *y[n-2]*. Also, *x[n]* is the current sample of input, while *y[n]* is the current output sample. The number of belated samples is defined by order of filter n where n = 2. Fig. 3 illustrates a graphical plot of noisy and filtered $y_2(t)$ signal together with spectrum of filtered $y_2(t)$ signal generated using the Fast Fourier Transform (FFT).

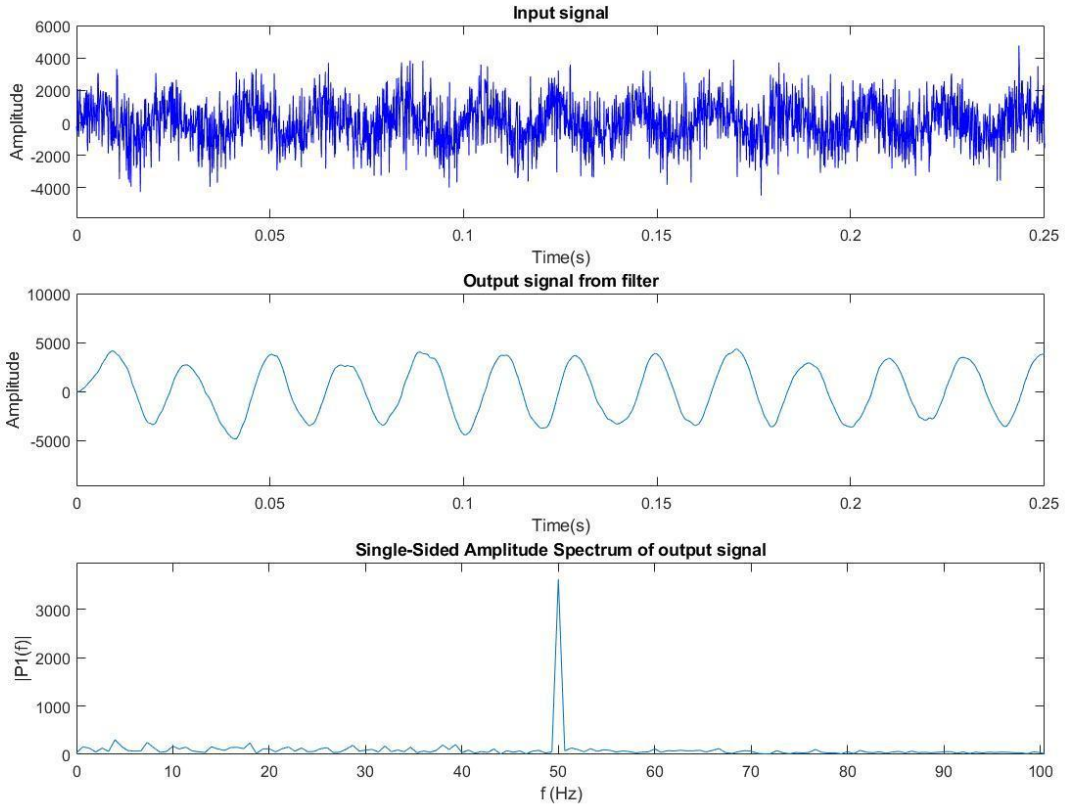


Figure 3. Graphical representation of noisy signal, filter output and single-side amplitude spectrum of output signal in MATLAB

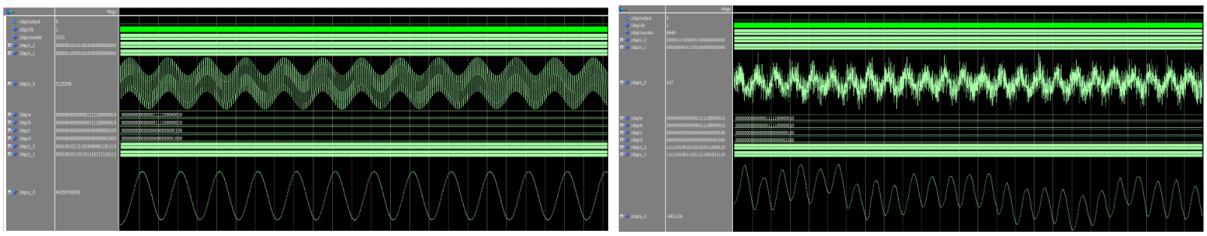
Full digital implementation of designed IIR filter whose main purpose is to filter out noise at higher frequencies (>>1 kHz) and preserve the low frequencies needs to be coded in VHDL for the execution on FPGA board. As opposed to the execution of only one instruction at a time as in Central Processing Unit (CPU), all the combinatorial logic executes simultaneously on the FPGA. For instance, a simple statement X = Y + Z generates on FPGA board by creating schematic composed of wiring up connections from Y to Z to an adder logic circuit and then connecting the wires from adder to X. In fact, digital design coded in VHDL causes the FPGA to wire up thousands of logic gates. This

FPGA implementation of the IIR filter takes about 2000 logic gates (adders, multipliers, and sequential logic circuits) wired together.

It begins by defining an input signal from ADC and output signal as *std_logic_vector* HDL type that goes into the Digital-to-Analog Converter (DAC) which further sends the filtered signal to the oscilloscope for display the signal. The Nyquist theorem is satisfied since sampling frequency is at least twice higher than max frequency available in a signal spectrum. An entity in VHDL is used to describe the interface of the designed filter, whilst architecture represents the behavioral description of the designed IIR filter. Further, it is required to declare and initialize the clock signal array (look-up table) composed of 20 000 samples coming from 12-bit ADC and counter which serves as index of the LUT. Afterwards, belated samples *x[n-1], x[n-2], y[n-1]* and *y[n-2]* as well as input/output samples *x[n], y[n]* together with calculated coefficients *a, b, c* and *d* are declared as *sfixed* HDL type (signed binary with fixed-point precision). In test bench simulation HDL *sfixed* type was used to store both the integer and the fraction part in the same signal. For actual testing on the FPGA board all signals are declared as a signed binary HDL type. There are 15 bits reserved for the integer part (bits indexed with positive numbers) and 12 bits reserved for fraction part (bits indexed with negative numbers) of signed binary number, except *y[n]* which has 34 bits for the integer part and 24 bits for fraction because of multiplication in difference equation between belated samples. For instance, the first coefficient value a = 1.938 is initialized and stored as 000000000000001|.|111100000010 fixed-point binary vector. Other signals used are initialized with all zeros, The first VHDL process in the test bench represents clock generation of 100 μ which oscillates between 0 and 1. Precisely, first 0 is assigned to clock, then after waiting for 50 microseconds, 1 is assigned to the clock, and this VHDL process constantly repeats.

The LUT is the most common method to approximate trigonometric functions on FPGA. The samples stored in the LUT table are of integer format. All samples are cast to *sfixed*/signed binary HDL type and assigned to the signal *x[n]* one by one. The integer and fraction part of the fixed-point binary vector are assigned separately. Additionally, the negative integer numbers are stored as 2's complement binary format where the most significant bit (MSB) is always 1. For example, -1036 samples are shifted right by 15 (integer part) and shifted left by 12 (fraction part), then all bits are inverted and added with a binary one. In this particular example, the fraction part has all zeros described in 2's complement format. The second VHDL process is called a clocked process because it executes on any change of the clock value. As a part of the clocked process if-conditional statement is added executing only with the rising edge of the clock (instant of time) from 0 to 1 but not oppositely. Thus, every signal assignment in this clocked process becomes register level design (flip-flop) meaning that values are updated with one clock later. In this clocked process, the same principle as in MATLAB is established including constant iteration over 20 000 samples available in the LUT. Specifically, 27 out of 57 bits (14 downto -12) of calculated output *y[n]* are assigned to the belated sample *y[n-1]*, and all other assignments are coded to store the older value of input/output signal. However, the calculation of difference equations, as well as the allocation of samples from LUT, are coded using concurrent assignment outside the second process in order to preserve overflow. Thus, these two statements execute instantaneously when anything on the right side of the assignment statement changes. This indicates the generation of combinatorial logic circuits during the synthesis of created FPGA-based digital design. Fig .4 illustrates two resulting simulations after compiling

and executing VHDL code of the IIR filter. Both $y_1(t)$ – composite signal of two different frequencies and $y_2(t)$ – signal with randomly generated noise are plotted in ModelSim environment using simulation time 100 microseconds.



1. Figure 4. a) ModelSim output of composite signal $y_1(t)$ of two different frequencies 50 Hz and 1 kHz b) ModelSim output of signal $y_2(t)$ with randomly generated noise

4. Result and analysis

The transfer function of the designed IIR filter was converted from CT to DT domain by calculating proper coefficients of a difference equation. After deriving an adequate difference equation of the system, it is further tested in MATLAB and implemented on the FPGA DE0-Nano board with direct VHDL coding. According to the Quartus 18.1 report, the total number of utilized logic elements is equal to 348 including 231 combinatorial logic circuits and 161 dedicated logic registers. The Register Transfer Level (RTL) diagram is shown in Fig .5. The input clock of the FPGA board is equally distributed to all registers which stored the belated samples and calculation of difference equations in each iteration. Every sample coming from ADC is directly stored into the register X_1 which is *D-Flip-Flop*. This register takes a value that is presented on the *D*-input at the rising edge of the clock signal and stores data on the output *Q*. It keeps stored value until the next rising edge of the clock signal. This clock signal is used for every generated register. The derived difference equation (19) indicates five multipliers, coefficients *c* and *d* are essentially considered as one and RTL design is described with only two multipliers including four adders. All adders and multipliers perform basic mathematical operations in parallel.

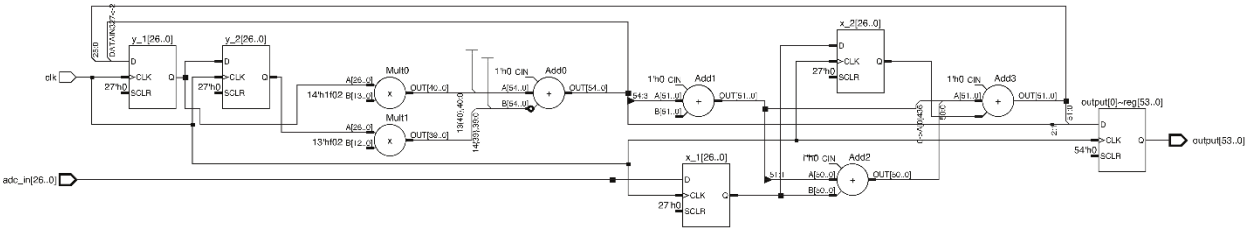


Figure 5. The RTL diagram of designed general-purpose IIR filter in Intel Quartus Prime for direct implementation on the FPGA board

Phase shift or phase distortion is presented in the second-order IIR filter and it can be noticed in Fig .4 where ModelSim simulation outcomes are depicted for both $y_1(t)$ and $y_2(t)$ signals. However, the bode plot shown in Fig .6 is the more

suitable approach to illustrate phase distortion of IIR filter for both cases namely, out-of-bound, and random noise signal.

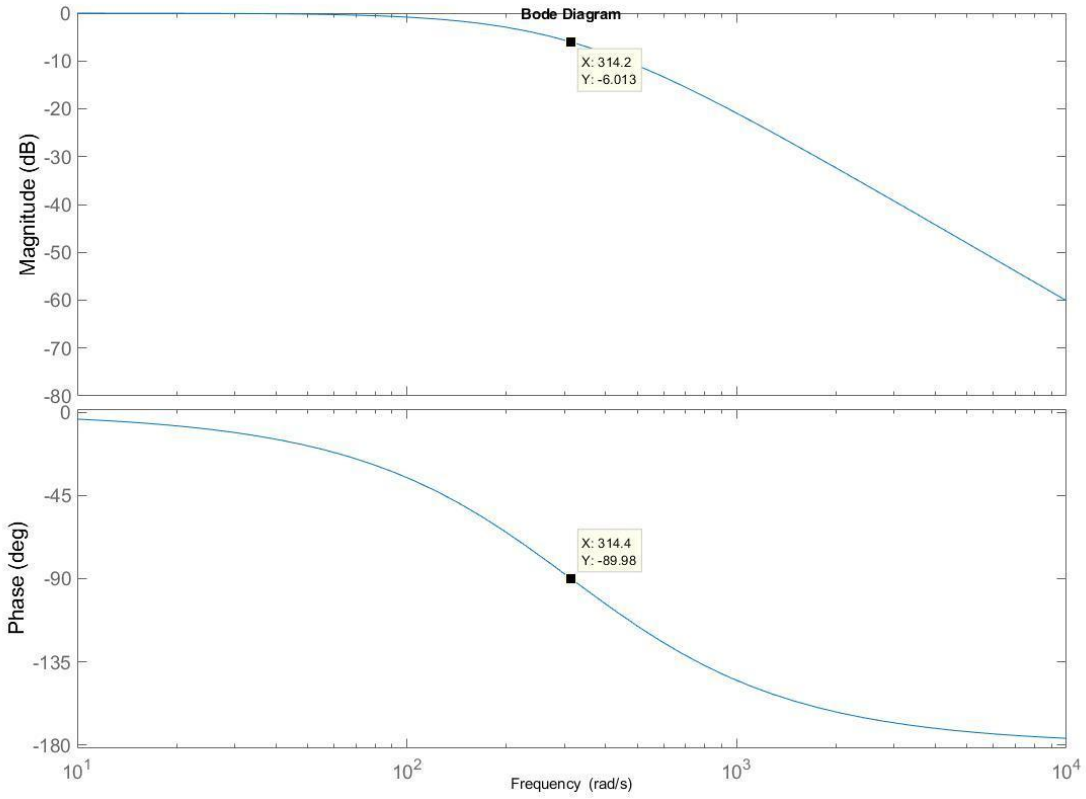


Figure 6. The Bode plot of the system indicating noise appearance in frequencies approximately larger than 1 kHz

Data available in the Bode plot show that for input signals with a frequency of 50 Hz, signal magnitude is decreased at approximately 6 dB, and it is decreasing for 40 dB/ decade. In addition to that, phase distortion is clearly visible on the graph. For input data of 50 Hz, phase distortion is -90^o. In this case, phase distortion does not affect the primary application of the designed filter since the main goal is to clear the noise from the signal. In some special cases, when the linear phase is required, FIR filters are a better option than IIR. Noise from the signal (with frequency of 1 kHz) is almost completely removed from the spectrum – magnitude is decreased for around 50 dB, which is enough to show the correct and desired operation of the designed filter.

5. Conclusion

This research study describes the complete digital implementation of IIR filter on the FPGA board. During this FPGA-based implementation, the major challenge was to adequately represent coefficients of the derived difference equation of the IIR filter since these values were floating-point. In case when linear distortion is not required, the IIR filters can be designed with much smaller order compared to FIR filters for almost the same system specification. The further research would include testing the performance of designed filters on bioelectrical signals such as EEG, ECG and other similar signals applied in medical purposes.

REFERENCES

[1] D. R. Wilson, D. R. Corrall, & R. F. Mathias, (1973). The Design and Application of Digital Filters. IEEE Transactions on Industrial Electronics and Control Instrumentation, IECI-20(2), 68–74.

[2] B. Farhang-Boroujeny, "A square-root Nyquist (M) filter design for digital communication systems," IEEE Trans. Signal Process., vol. 56, no. 5, pp. 2127-2132, May 2008.

[3] Z. Cheng-liang, and A. H. Wang. "IIR Digital Filter Design Research and Simulation on MATLAB."

[4] M. B. Trimale, and Chilveri. "A review: FIR filter implementation". 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT).

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

[6] S. A. Ito and L. Carro, "A comparison of microcontrollers targeted to FPGA-based embedded applications," Proceedings 13th Symposium on Integrated Circuits and Systems Design (Cat. No.PR00843), Manaus, Brazil, 2000, pp. 397-402.

[7] K. Sridharan, and T.K. Priya, "The Design of a Hardware Accelerator for Real-Time Complete Visibility Graph Construction and Efficient FPGA Implementation," Industrial Electronics, IEEE Trans. Ind. Electron., vol. 52, n°4, pp. 1185-1187, Aug. 2005.

[8] C. A. Fields. "Proper use of hierarchyin HDL-based high density FPGA design". In Will Moore and Wayne Luk, editors, Field-Programmable Logic and Applications: 5th International Workshop, FPL '95, volume 975 of Lecture Notes in Computer Science, pages 168–177, Berlin, Germany, August 1995. Springer Verlag.

[9] R. Sarić, D. Jokić, and N. Beganović "Implementation of Neural Network-Based Classification Approach on Embedded Platform". In: Badnjević A., Škrbić R., Gurbeta Pokvić L. (eds) CMBEBIH 2019. CMBEBIH 2019. IFMBE Proceedings, vol 73. Springer, Cham.

[10] D. G. Manolakis, "Applied Digital Signal Processing, Theory and Practice, (Massachutetts Institute of Technology Lincoln Labaratory), Vinay K. Ingle (Northeastern University, Boston).

[11] Y. Liao, & X. Wang, "General Rules of Using Bode Plots for Impedance-Based Stability Analysis", (2018) IEEE COMPEL 2018 At: Padova, Italy.