An approach to Digital Low-Pass IIR Filter Design

Bojan Jovanović, and Milun Jevtić

Abstract – The paper describes the design process of discrete network – digital low-pass filter with Infinite Impulse Response (IIR filter). Based on given parameters that network should meet, and with the use of HDL hardware description language and MATLAB software package, hardware implementation of the filter transfer function was designed. Simulations confirmed the validity of the implementation design in an FPGA chip.

Keywords – Digital low-pass IIR filter, VHDL, MATLAB.

I. Introduction

Digital processing of continuous signal is based on numerical processing of the data which are used to present continuous signals. It has its beginning in the first numerical methods for solving mathematical problems such as numerical solution of integral-differential equations, numerical solution of integrals, interpolation etc. With the development of the computing machines already existing algorithms for digital signal processing are applied. New algorithms are also developed. Fig. 1 shows a simplified block diagram of digital processing of continuous signals [1]. Continuous signal that is processed is marked with *x(t)*. After analog-to-digital conversion in A/D block we get the digital signal *{x(n)}* which is presented in the form of an array of numbers. This signal is then processed using some processing device (DSP, FPGA, CPLD etc.). We will adopt that *x(n)*, which represents the output from A/D converter, is 8-bit binary number represented in second complement. Signal from the output of processing device is also an array of numbers *{y(n)}*. This array is converted into continuous signal *y(t)* using D/A block for digital-to-analog conversion. Each member of the array *{y(n)}* is also 8-bit binary number represented in the second complement.

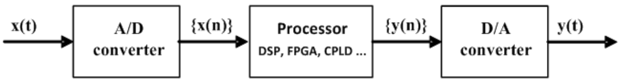


Fig. 1. Digital processing of continuous signals

Digital signal processing has a very broad application areas, ranging from simulation of analog (continuous) networks on digital computers to the development of new digital systems that will completely replace analogue.

There are many advantages of digital over analog signal processing. Digital signal processing can be realized with higher degree of accuracy. Then, one computer can simultaneously perform multiple digital signal processing's. Digital networks are very flexible, easily can be changed by changing the program parameters. Significant advantage is also in the fact that the computer can execute a series of transformations that can not be achieved using analogue elements, either because of the complexity of the transformations, either due to lack of analogue elements required for transformation. The changes of network parameters due to aging of components does not exist in digital networks. Characteristics of digital networks are stable, and reliability is great. We should not neglect the fact that the price of digital components is significantly lower than the cost of analog components. In terms of physical dimensions of the networks the advantage is also on the digital processing side.

The limitation of digital networks is that the signal processing takes some time. Also, the use of digital networks for processing high frequency digital signals is limited. Besides the limitations related to the speed of digital signal processing digital networks have more drawbacks, such as the existence of the noise and low dynamic of the signal. Noise can not be avoided because it occurs as a result of signal quantisations, rounding of the product and rounding of the sum in the numerical calculations. Noise reducing and increasing of the signal dynamic can be achieved by increasing the length of digital words which are used to represent numbers. However, the speed of filter reduces with digital word length increasing. It is therefore necessary to find compromise between the filter speed and the size of the noise generated by digital network [2].

II. Procedure of designing recursive IIR filter

A. Sampling Theorem

According to sampling theorem, frequency limited continuous signal *x(t)* can be reconstructed from its discrete samples if the sampling frequency is at least *Fs=2fc*, where *fc* is the cutoff frequency in continuous signal frequency spectrum. Analog signals (audio, video etc.) can therefore be processed digitally and the processing result can be converted to analog domain. One of the key requirements for correct processing of analog signal *x(t)* is the frequency of its sampling *Fs*. Sampling frequency must be at least twice the highest frequency from the frequency spectrum of signal *x(t)*. To properly reproduce 20kHz audio signal it must be sampled at least 40.001 times per second (20.000 x 2 + 1).

Bojan Jovanović and Milun Jevtić are with the Department of Electronics, Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia,
E-mail: bojan@elfak.ni.ac.rs. milun.jevtic@elfak.ni.ac.rs

B. Designing task

Design digital low-pass filter with Infinite Impulse Response and the following characteristics:

- Sampling frequency: $F_s=44kHz$;
- Maximal passband frequency: $f_p=5kHz$;
- Minimal stopband frequency: $f_a=8kHz$;
- Maximal signal attenuation in passband $a_{max}=1dB$;
- Minimal signal attenuation in stopband $a_{min}=50dB$;
- For referent filter adopt Chebyshev approximation.

Filter attenuation characteristics are given in Fig. 2.

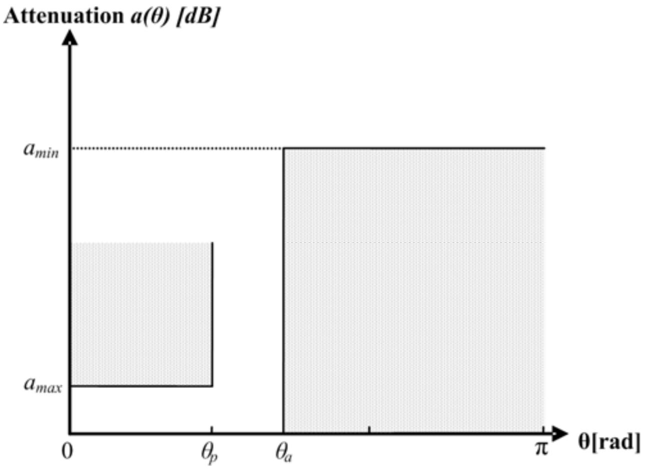


Fig. 2. Attenuation characteristic of low-pass filter

C. Designing procedure

Fig. 2 shows filter attenuation characteristic in respect to discrete frequency θ which is in following relation with analog frequency f:

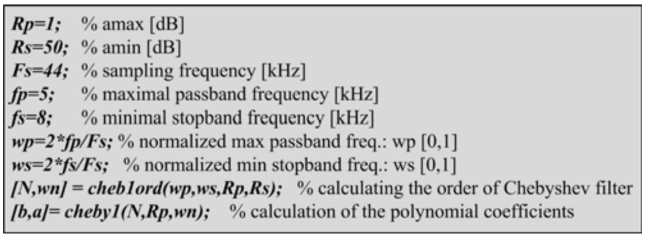
$$\theta = \frac{2\pi f}{F_s} \qquad (1)$$

where Fs is sampling frequency.

For discrete frequency $\theta=\pi$ analogue frequency is $f=F_s/2$. According to sampling theorem this is the maximal frequency in frequency spectrum of signal $x(t)$. Therefore we consider attenuation characteristic for discrete frequency range [0, π]. Passband frequency range is [0, θ_p] and maximal signal attenuation in this range is $a_{max}=1dB$. Stopband frequency range is [θ_a, ∞] and minimal signal attenuation in this range is $a_{min}=50dB$. Frequency range [θ_a, θ_p] represents transition zone. There are not defined tolerances of attenuation characteristic.

Based on given filter parameters, and using MATLAB software package [3] we calculate the filter's order and the polynomial coefficients of filter transfer function in z-domain – $H(z)$ (Fig. 3).

$$H(z) = \frac{B(z)}{A(z)} \qquad (2)$$



```
Rp=1;    % amax [dB]
Rs=50;   % amin [dB]
Fs=44;   % sampling frequency [kHz]
fp=5;    % maximal passband frequency [kHz]
fs=8;    % minimal stopband frequency [kHz]
wp=2*fp/Fs; % normalized max passband freq.: wp [0,1]
ws=2*fs/Fs;  % normalized min stopband freq.: ws [0,1]
[N,wn] = cheb1ord(wp,ws,Rp,Rs);  % calculating the order of Chebyshev filter
[b,a]= cheby1(N,Rp,wn);   % calculation of the polynomial coefficients
```

Fig. 3. MATLAB function for filter order and polinomial coefficients calculation

After calculation in MATLAB, for filter order is obtained $N=7$, and filter transfer function coefficients are:
b= 10^{-3}*[0.0176 0.1232 0.3696 0.6160 0.6160 0.3696 0.1232 0.0176];
a = [1.0000 -5.5152 13.7614 -20.0229 18.2902 -10.4726 3.4791 -0.5178].

For filter transfer function we have the following equation:

$$H(z) = \frac{b_0 z^7 + b_1 z^6 + \ldots + b_6 z^1 + b_7}{a_0 z^7 + a_1 z^6 + \ldots + a_6 z^1 + a_7} \qquad (3)$$

Stability condition for discrete networks is that all the poles of network transfer function lie within unity circle in z-plane. MATLAB function ***zplane(b,a)*** calculates and displays in z-plane zeros and poles of filter transfer function (Fig. 4).

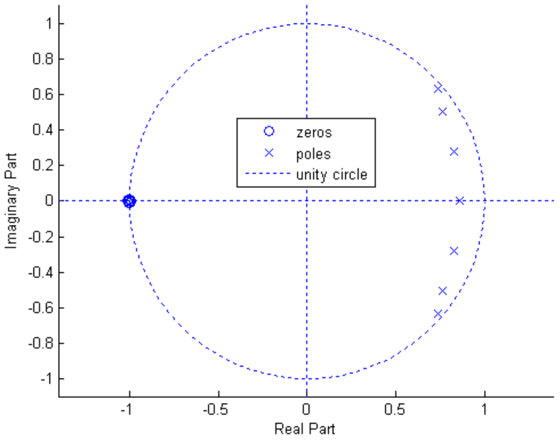


Fig. 4. Zeros and poles of filter transfer function

From Fig. 4 can be seen that discrete network meets stability requirement.

The next step is quantization of coefficients of the polynomial transfer function. Coefficients quantization means their presentation with finite number of bits.

Considering that some coefficients are negative and the maximal absolute value of coefficients is 20.0229, to represent the integer part we need a minimum of 6 bits. It is now needed to determine the minimum number of bits to represent fractional part so that filter attenuation characteristics *a(θ)[dB]* do not exceed beyond given tolerances. Filter attenuation is defined as the amplitude characteristic of the filter transfer function *H(z)* on the unit circle and it is expressed in decibels:

$$a(\theta) = -20\log\left|H(e^{j\theta})\right| \tag{4}$$

Fig. 5 shows comparative characteristics of the filter attenuation in case of quantized and unquantized coefficients of polynomial of transfer function. Quantized coefficients are represented with 17 bits, in the format [17 11], 6 bits for integer, and 11 bits for fractional part.

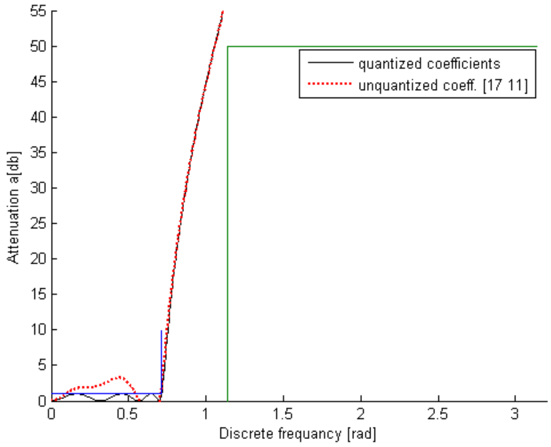
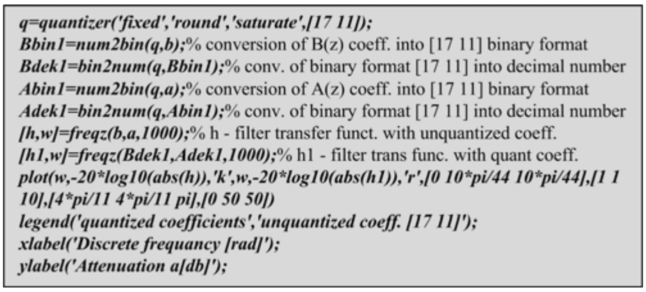


Fig. 5. Attenuation characteristic for quantized [17 11] and unquantizad coefficients

A set of MATLAB commands used for calculate and plot comparative characteristics is shown in Fig. 6.



```
q=quantizer('fixed','round','saturate',[17 11]);
Bbin1=num2bin(q,b);% conversion of B(z) coeff. into [17 11] binary format
Bdek1=bin2num(q,Bbin1);% conv. of binary format [17 11] into decimal number
Abin1=num2bin(q,a);% conversion of A(z) coeff. into [17 11] binary format
Adek1=bin2num(q,Abin1);% conv. of binary format [17 11] into decimal number
[h,w]=freqz(b,a,1000);% h - filter transfer funct. with unquantized coeff.
[h1,w]=freqz(Bdek1,Adek1,1000);% h1 - filter trans func. with quant coeff.
plot(w,-20*log10(abs(h)),'k',w,-20*log10(abs(h1)),'r',[0 10*pi/44 10*pi/44],[1 1
10],[4*pi/11 4*pi/11 pi],[0 50 50])
legend('quantized coefficients','unquantized coeff. [17 11]');
xlabel('Discrete frequancy [rad]');
ylabel('Attenuation a[db]');
```

Fig. 6. MATLAB code for calculating and plot filter attenuation characteristics

From Fig. 5 can be seen that the attenuation characteristic with quantized coefficients exceeds outside the given tolerances for passband maximal attenuation, ie. that attenuation in passband is greater than *1dB*. Therefore, for the presentation of fractional part one more bit is added. Quantized coefficients are now in the format [18 12]. Fig. 7 shows comparative characteristics of the filter attenuation in case of quantized (in the format [18 12]) and unquantized coefficients.

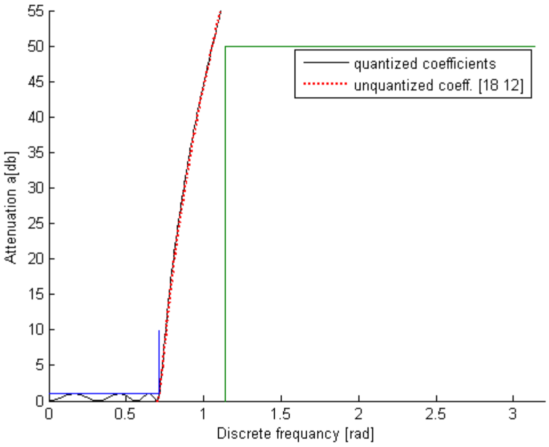


Fig. 7. Attenuation characteristic for quantized [18 12] and unquantizad coefficients

Fig. 7 shows that with quantization in the format [18 12] we obtain filter attenuation characteristic that meets the given tolerances both in passband and in stopband.

In addition to the direct realization of the transfer function, there are cascade, parallel, scale and grid implementations.

Cascade realization is based on factorization of digital filter transfer function into the product of transmission functions of the first and second order:

$$H(z) = G\prod_{i=1}^{K} H_i(z) \tag{5}$$

where $H_i(z)$ are the transfer functions of the first:

$$H_i(z) = \frac{1 + b_{1i}z^{-1}}{1 + a_{1i}z^{-1}} \tag{6}$$

or second order:

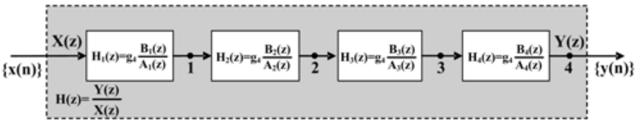
$$H_i(z) = \frac{1 + b_{1i}z^{-1} + b_{2i}z^{-2}}{1 + a_{1i}z^{-1} + a_{2i}z^{-2}} \tag{7}$$

K is the number of sections and *G* is the constant.
Given that our obtained filter transfer function is of the seventh row, it breaks on three second order and one first order transfer functions.

MATLAB code ***[sos,g]=tf2sos(b,a);*** is used to get the polynomial coefficients of first and second order filter transfer functions. These coefficients are:

b1 = [1.0000   1.0101]
a1 = [1.0000 -0.8577]
b2 = [1.0000   2.0126   1.0127]
a2 = [1.0000 -1.6544   0.7640]
b3 = [1.0000   1.9955   0.9956]
a3 = [1.0000 -1.5325   0.8390]
b4 = [1.0000   1.9818   0.9819]
a4 = [1.0000 -1.4706   0.9418]

Constant *G* has the value of $G=1.76x10^{-5}$. It can be noted that the coefficients of these polynomials are considerably smaller than polynomial coefficients of transfer function in the case of direct implementation. Fig. 8 shows the block diagram of digital filter cascade realization.



X(z)
{x(n)}
$H_1(z)=g_4\frac{B_1(z)}{A_1(z)}$
1
$H_2(z)=g_4\frac{B_2(z)}{A_2(z)}$
2
$H_3(z)=g_4\frac{B_3(z)}{A_3(z)}$
3
$H_4(z)=g_4\frac{B_4(z)}{A_4(z)}$
4
Y(z)
{y(n)}
$H(z)=\frac{Y(z)}{X(z)}$

Fig. 8. Block diagram of cascade realization of digital filter transfer function

Filter structure is the cascade connection of one first order and three second order sections. Constant *G* is evenly distributed in all sections:

$$g_4 = \sqrt[4]{G} = 0.0648 \tag{8}$$

In cascade realization there are problems of constant G distribution among the sections as well as the sequence of the sections. We will not consider them now.

In terms of used components direct and cascade realization are practically equivalent. However, in the case that digital filter need to be hardware realized, cascade realization is more convenient, having in mind its modularity.

Based on the transfer functions $H_i(z)$, $i=1,2,3,4$ we get differential equations in time domain, which are used during the hardware implementation of each module.

Differential equation for first order transfer function is:

$$\begin{aligned} y(n) &= 0.8577y(n-1) + 0.0648x(n) + \\ &+ 0.0655x(n-1) \end{aligned} \tag{9}$$

From the equation we can see that for the calculation of the output sequence *{y(n)}* n-th member we need to know prior calculated *y(n-1)* member and two members of the input sequence *{x(n)}*. Constants that multiplies these elements are represented in the second complement, in the format [18 12]. Shown in Fig. 9 is the block diagram of the first order section hardware implementation. Section consists of data path and control unit [4], [5].

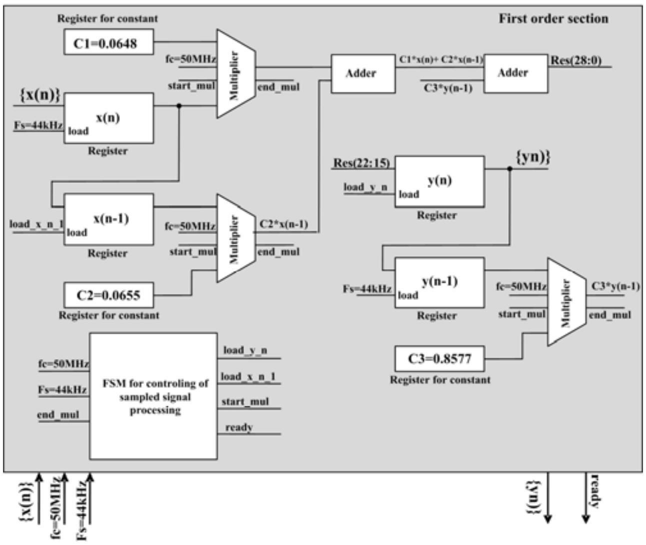


Fig. 9. First order section block diagram

The input to the first order section is sampled signal *{x(n)}*, which, as it is said, is a sequence of 8-bit numbers represented in the second complement. Signal *{x(n)}* is the result of A/D conversion of analog signal *x(t)* which we want to filter. *x(n)* and *x(n-1)* elements of the input sequence *{x(n)}* are stored in 8-bit registers as well as elements of the output signal *{y(n)}*, *y(n)* and *y(n-1)*. Constants C_1, C_2 and C_3 are stored in 18-bit registers. Data path also includes adders and multipliers. Entire process of processing input signal elements is up to control unit – Finite State Machine (FSM). The adder is of the *ripple-carry* type, based on *full adders*. Multiplier has *start_mul* input signal that initiates the process of multiplication and the output signal *end_mul* which signals that the process of multiplication is over. Input signals to the first order section are also two clock signals: *clock1* with frequency of $f_c=50MHz$ which is used to synchronize the processing of sampled signal and *clock2* with frequency of $F_s=44kHz$ (sampling frequency) which synchronizes the storage in registers of elements *x(n)* and *y(n-1)*. From Fig. 9 can also be seen that register *y(n)* stores only 8 bits of result *Res(28:0)*. The first 15 bits of the result signal, *Res(14:0)*, are the fractional part, because the constant C_1 is in the format [18 12] and therefore the result of multiplication has 15 bits in fractional part. 6 MSB bits of the result, *Res(28:23)*, are a sign extension of 8-bit number that is stored in register *y(n)*. These 6 bits also aren't of importance.

Setting the output signal *ready* FSM indicate the end of sample *x(n)* processing. Now, 8-bit output *y(n)* has the valid data.

Differential equations for second order sections are:

$$\begin{aligned} &y(n) = 1.6544y(n-1) - 0.764y(n-2) + \\ &+ 0.0648x(n) + 0.1304x(n-1) + 0.0656x(n-2) \end{aligned} \tag{10}$$

$$y(n) = 1.5325y(n-1) - 0.839y(n-2) + \\ + 0.0648x(n) + 0.1292x(n-1) + 0.0645x(n-2) \tag{11}$$

$$y(n) = 1.4706y(n-1) - 0.9418y(n-2) + \\ + 0.0648x(n) + 0.1284x(n-1) + 0.0636x(n-2) \tag{12}$$

From the equations can be seen that for calculating the n-th member of the output sequence *{y(n)}* we need to know prior calculated *y(n-1)* and *y(n-2)* members and three members of input sequence *{x(n)}*. Constants that multiplies these elements are also represented in the second complement, in the format [18 12]. Shown in Fig. 10 is the block diagram of the second order section hardware implementation.

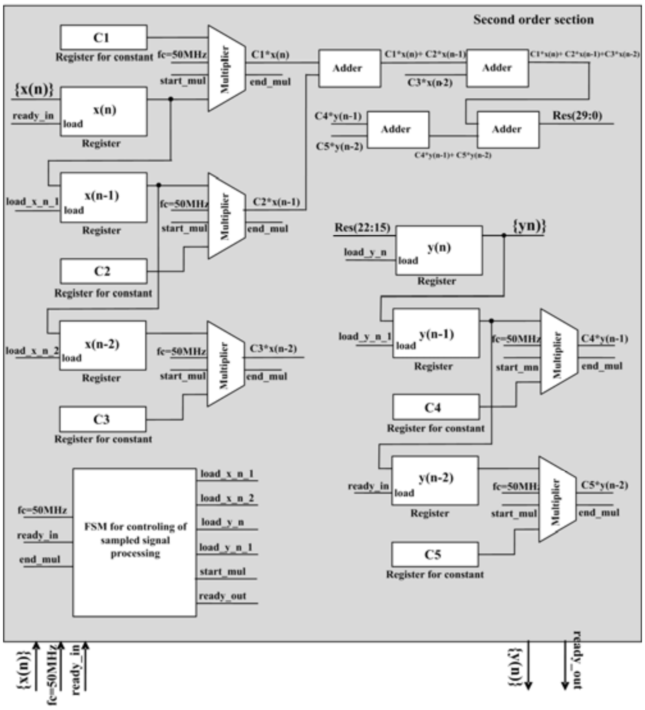


Fig. 10. Second order section block diagram

Second order section is organized as well as first order section. It also consists of data path and control unit. Unlike the first order filter section, this section has two more multipliers and two more adders as well as additional 8-bit and 18-bit registers for storing coefficients and *x(n-2)* and *y(n-2)* elements. For the same reasons as in first order section, register *y(n)* stores only 8 bits of the result *Res(29:0)*.

Finally, Fig. 11 shows the block diagram of low-pas IIR digital filter cascade realization.

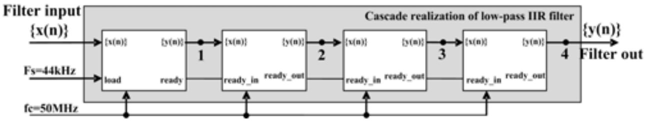


Fig. 11. Cascade realization of digital low-pass IIR filter

Each section of digital filter hardware realisation is described and simulated in Active-HDL software package, using shematic and text editor and VHDL language. Validation of filter is made for input signal:

$$x(t) = 2\sin(2\pi f_1 t) + 2\sin(2\pi f_2 t) \tag{13}$$

The input signal is sum of two sinusoids with frequencies $f_1=2kHz$ and $f_2=9kHz$. Amplitude of both sinusioids are $A=2$. Considering that the filter bandwidth is $f_p=5kHz$ we expect that sinusioid with frequency f_2 should be suppresed in the filtered output signal. Filtered signal therefore need to be sinusoid with frequency f_1 and amplitude $A=2$.
Numbers 1, 2, 3 and 4 in Fig. 11 indicate the outputs from the first, second, third and fourth filter section, respectively. Filter input *{x(n)}* is the sequence of 8-bit numbers represented in the second complement. This sequence is obtained on the output of A/D converter which converts analog voltages within a range of [-5, 5V] to 8-bit digital number presented in the second complement. A/D converter sampling frequency is $F_s=44kHz$.

Filter simulation is also performed in MATLAB. MATLAB code for this simulation is shown in Fig. 12.

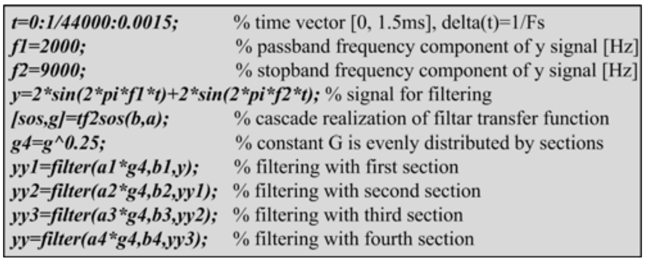

```
t=0:1/44000:0.0015;          % time vector [0, 1.5ms], delta(t)=1/Fs
f1=2000;                     % passband frequency component of y signal [Hz]
f2=9000;                     % stopband frequency component of y signal [Hz]
y=2*sin(2*pi*f1*t)+2*sin(2*pi*f2*t); % signal for filtering
[sos,g]=tf2sos(b,a);         % cascade realization of filtar transfer function
g4=g^0.25;                   % constant G is evenly distributed by sections
yy1=filter(a1*g4,b1,y);      % filtering with first section
yy2=filter(a2*g4,b2,yy1);    % filtering with second section
yy3=filter(a3*g4,b3,yy2);    % filtering with third section
yy=filter(a4*g4,b4,yy3);     % filtering with fourth section
```

Fig. 12. MATLAB code for signal filtering

At least, Figs. 13 and 14 show comparative results of filter simulation. Fig. 13 shows simulation results using Active-HDL and Fig. 14 simulation results using MATLAB software packate.

As can be seen, there is some degree of mutual match of simulation result. Mutual match would be even better if we reduce digital noise ie. if elements of input sequence *{x(n)}* were presented with more than 8 bits. That means the use of better A/D converter for analog-to-digital signal conversion. Also, mutual match would be better if the constants were presented in a larger than [18 12] format.

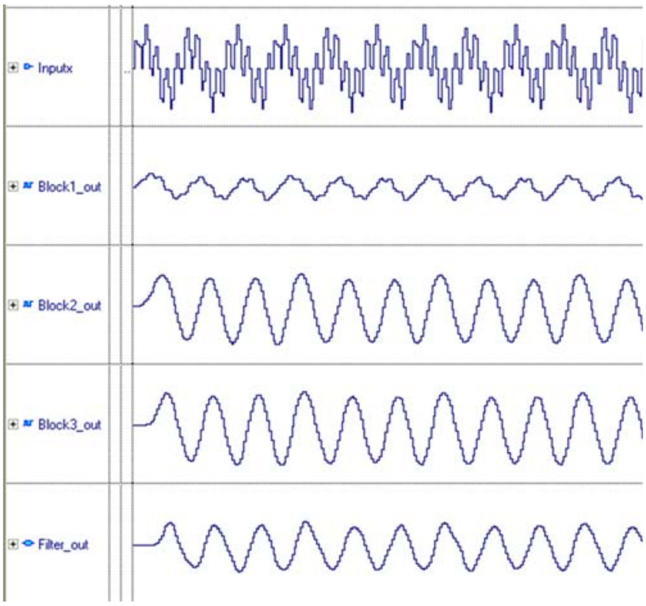


Fig. 13. Filter simulation in Active-HDL software package

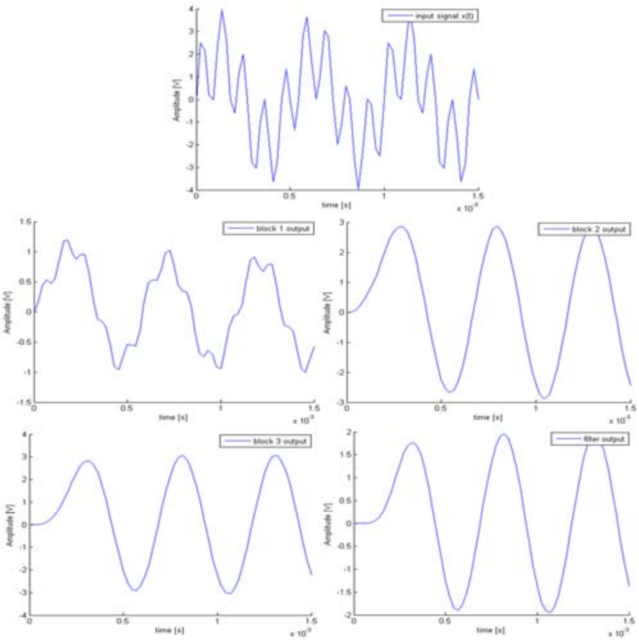


Fig. 14. Filter simulation in MATLAB software package

IV. Conclusion

HDL description of digital low-pass IIR filter can be implemented in any programmable logic component (CPLD or FPGA) that has the appropriate number of programmable logic blocks. Since each section consists of data path and control unit it is possible to perform section optimization by the criteria of size, speed, power consumption, etc.

References

[1] Stojanović, V., “*Diskretne mreže i procesiranje signala*”, Elektronski fakultet, Niš, 2004.

[2] Winder, S., “*Analog and Digital Filter Design*”, Supertex Inc., Ipswich, 2002.

[3] Ingle, V., Proakis, J., “*Digital signal processing using MATLAB, 2e*”, Thomson, Boston, 2007.

[4] Pedroni, V., “*Circuit Design with VHDL*”, MIT Press, London, 2004.

[5] Ronald, T., Neal, W., “*Digital Systems-Principles and applications*”, Prentice Hall, New Jersey, 2007.