IMPLEMENTATION OF DIGITAL FILTERS IN FPGA STRUCTURES

Mózes Ferenc-Emil, Germán-Salló Zoltán
Petru Maior University of Tirgu-Mureş
mozes.ferenc.emil@gmail.com, zgerman@engineering.upm.ro

ABSTRACT

Filtering is the most common task in digital signal processing, when some particular parameters of the input signal are removed or modified and a new signal is obtained at the output. The most common filtering goal is to remove noise from a signal. Digital filtering deals with discrete sequences, their implementation require both soft and hard solutions. This work focuses on some useful methods to implement digital filters in reconfigurable hardware structures, as FPGAs. This paper presents FIR filters of different order and structures, and using the MathWorks® tools, shows how these structures can be mapped to the Xilinx® FPGA architecture. The implemented digital filters are evaluated through signal to noise ratio and root mean square error between the filtered and the noisy signal.

Keywords: digital filter, FPGA, FIR filter

1. Introduction

Filtering is a fundamental aspect of signal processing which performs direct manipulations on the frequency band of signals, removing undesirable parts as noise or extracting useful components. There are two main types of filter, analog and digital, they are very different in physical structure and function. An analog filter uses analog electronic circuits made up from components such as resistors, capacitors and operational amplifiers to produce the required filtering effect. A digital filter uses a digital processor to perform numerical calculations on sampled values of the signal. They can achieve better performance than analog filters (limited by the accuracy of their electronic components), can easily be changed without affecting the circuitry (hardware). An analog filter can only be changed by redesigning the filter circuit. Digital filters are build from three basic elements: an adder, a multiplier and a delay element and can be implemented using either a block diagram or a signal flow graph. Recently the Field Programmable Gate Arrays (FPGAs) became one of the most preferred platform for evaluating and implementing signal processing algorithms. They have special features, like embedded multipliers, dedicated DSP circuitry, scalability and re-configurability therefore they are one of the most attractive platforms for advanced signal processing algorithms [1, 6]. Some engineering tools like MATLAB have enabled the design of various digital filter structures in a faster and more accurate way with the possibility of implementation in FPGAs. The second part of this work presents briefly the most common digital filter structures and some possibilities to implement them. The proposed implementation and structures are shown in the third part. Experimental results obtained with different structures on test signal are presented in the fourth section. Finally, the conclusions show some possible directions for further work.

2. Digital filter realizations

The Fourier transform theory says that the linear convolution of two sequences in the time domain is the same as multiplication of two corresponding spectral sequences in the frequency domain. Therefore filtering is in the multiplication of the signal spectrum by the frequency domain impulse response of the filter. Usually a digital filter is a simple discrete-time, discrete-amplitude convolver which performs a convolution of the time domain impulse response and the discrete (sampled) signal. There are two types of digital filters: a non-recursive (known as an FIR or Finite Impulse Response filter) and a recursive (named IIR or Infinite Impulse Response) filter [3, 4]. An FIR type has the output as a convolution between the input sequence and the impulse response of filter:

$$y(n) = \sum_{k=0}^{N-1} b(k) \cdot x(n-k) \qquad (1)$$

A large percentage of filters implemented in the digital domain are Finite Impulse Response (FIR) filters. These filters are used over a wide range of sample rates and are supported in terms of software tools, and FPGA IP cores. An FIR filter is usually

implemented by using a series of delays, multipliers, and adders in order to have the desired filter's output.

Another type of digital filter is the Infinite Impulse Response (IIR) filter, which has the output

$$y(n)=\sum_{k=0}^{N-1}a(k)\cdot x(n-k)+\sum_{k=0}^{M-1}b(k)\cdot y(n-k) \qquad (2)$$

As it can be seen, portions of the output are returned to be recomputed "over and over" by the same few coefficients, meaning that IIR structure uses feedback [3, 4].

Structurally, FIR filters consist of two elements: a sample delay line and a set of coefficients. To implement the filter, the input sample is put into the delay line, each sample is multiplied by the corresponding coefficient and the result is accumulated, after that the delay line is shifted by one sample to make room for the next input sample [6]. This is the serial direct form, presented in fig. 1.

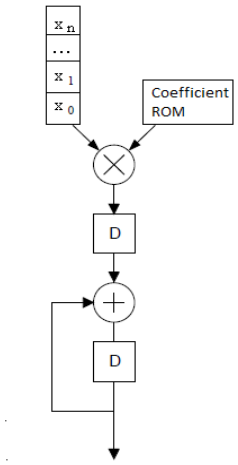

Fig. 1 – Serial Direct Form FIR filter

Another implementation structure called the transposed form FIR filter, where data samples are applied in parallel to all the tap multipliers through pipeline registers is shown in fig. 2. The products are applied to a cascaded chain of registered adders.

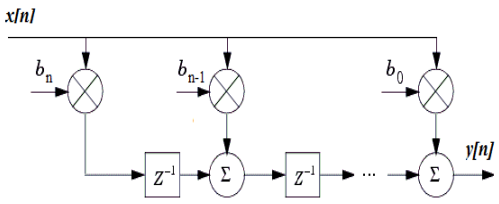

Fig. 2 – Transposed FIR filter

3. The Implementation procedure

The Filter Design and Analysis Tool (FDA Tool) is a graphical user interface (GUI) available in the Signal Processing Toolbox of MATLAB for designing and analyzing filters. The FDA Tool uses double precision floating point representation for the design calculations. This allows the tool to achieve a fair degree of precision, which is reflected in the close-to-ideal response of the reference filter. This tool along with the Simulink HDL Coder is a powerful utility to generate HDL versions of the designed filters.

In the first place a model only using Simulink's common blocks and the FDATool block from the Signal Processing Toolbox was designed. This can be seen in fig. 3.

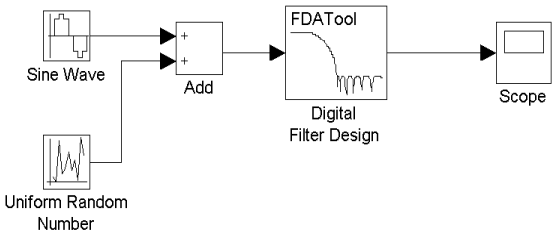

Fig. 3 – Filter simulation model

To evaluate the performances of the designed filter model, a sine wave was used as a test signal with known added noise. This made possible to compute the signal to noise ratio and filtering error. This model was later built using Xilinx's System Generator blocks from Simulink, created with the purpose to be transferable to FPGAs [5].

Five low-pass filters of different order (8, 12, 16, 20, 24) were designed using FDA Tool, with the following parameters: F_s = 4 kHz, F_{pass} = 100 Hz, F_{stop} = 300 Hz. Figure 4 presents the specifications of the 24th order filter, i.e. the magnitude and phase response in the specified frequency domain.

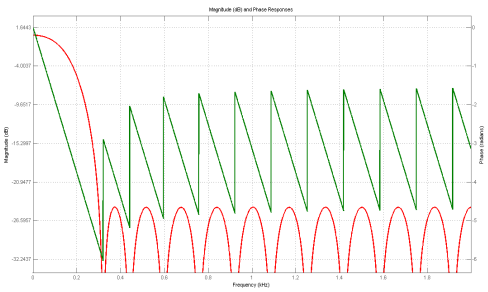

Fig. 4 – Magnitude and phase response of the 24th order filter

The designed filters were exported to VHDL models using the Simulink HDL Coder. The Black Box block from the System Generator blockset offered the possibility to simulate VHDL code within Simulink. Figure 5 shows the obtained model.

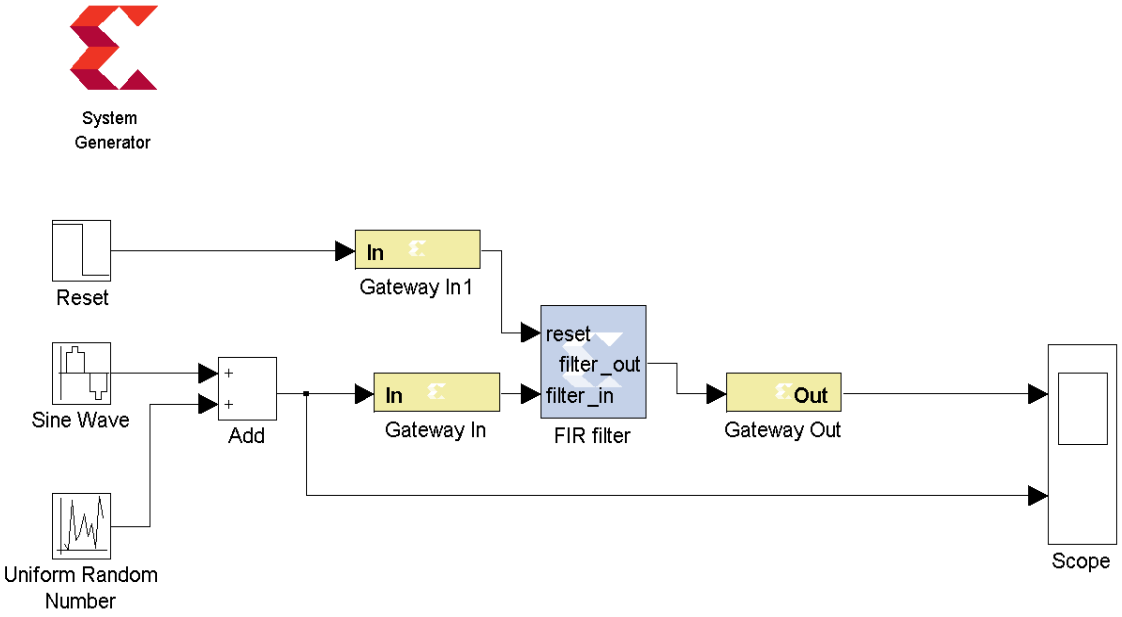

Fig. 5 – Simulink model of a FIR filter, using Xilinx's System Generator blocks

4. Results

A 100 Hz frequency sine wave corrupted with a uniform noise was considered as input for the five filters. The signal to noise ratio, the gain and the resulting error were calculated in case of every filter. Equation 3 was used to calculate the signal to noise ratio, while the error was obtained using equation 4.

$$SNR_{dB} = 10\lg\frac{P_{s_f}}{P_{s_f - s_n}} \tag{3}$$

$$\varepsilon = \frac{1}{N}\sum_{i=1}^{N}(s[i] - s_f[i])^2 \tag{4}$$

P_{s_f} denotes the power of the filtered signal, $P_{s_f - s_n}$ is the power of the difference between the filtered signal and the noisy signal (this difference is the estimated noise) and s is the original, uncorrupted sine wave.

Figure 6 shows the input signal and the output of the filters.

Table 1 Measurement results

Filter order	SNR [dB]	Error
8	7.0390	0.0780
12	11.5297	0.0592
16	11.7168	0.0573
20	12.4403	0.0569
24	12.8034	0.0561

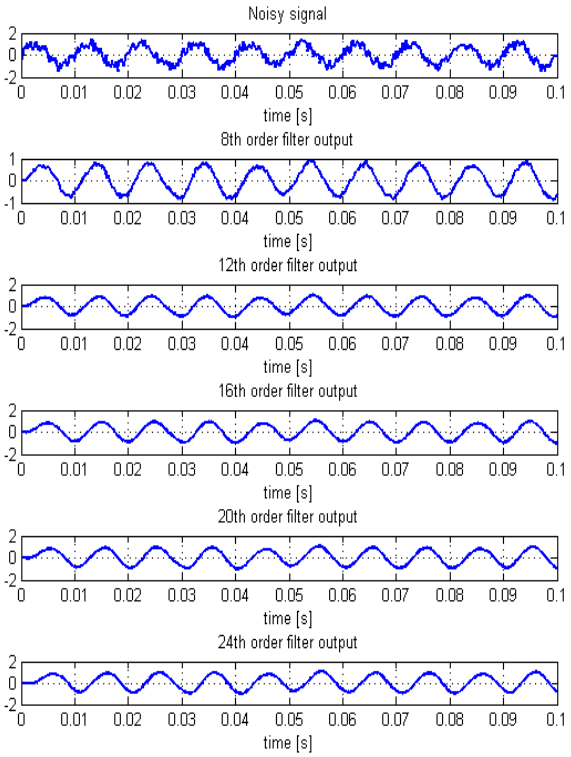

Fig. 6 – The original signal and the different filtered signals

Figures 7 and 8 were are based on the values from table 1. Figure 7 shows the values of the signal to noise ratios for each filter. It can be observed that the higher the filter order is, the higher the signal to noise ratio becomes.

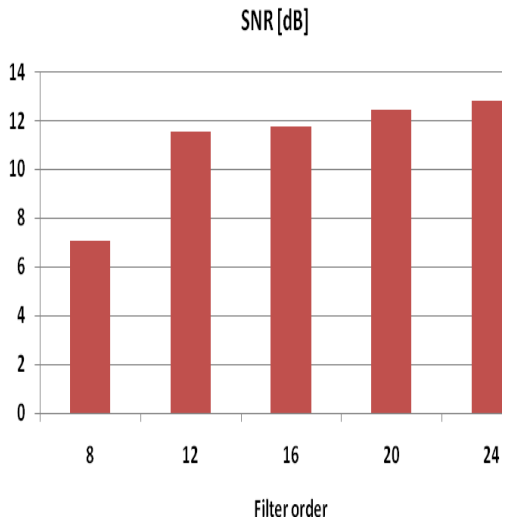

Fig. 7 – The obtained signal to nosie ratio values

Figure 8 shows the evolution of the error during the filtering process.

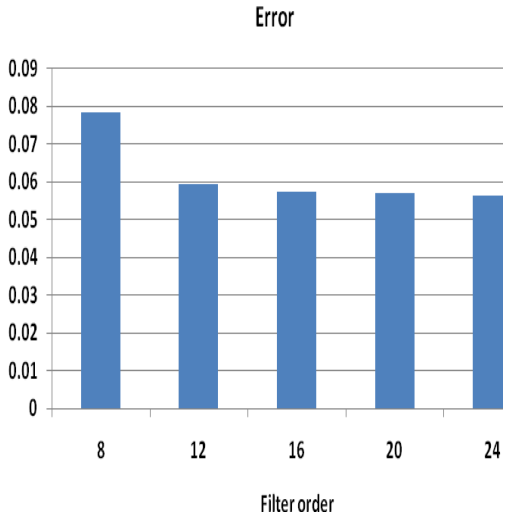

Fig. 8 – The obtained error for each of the filters

5. Conclusions

The filters were synthesized for the Spartan 3E FPGA produced by Xilinx. Table 2 contains the resource utilization statistics for the 24th order FIR filter.

Table 2 Resource utilization

Resource name	Used	Available	Percentage
Slice Flip Flops	961	17344	4 %
Slices	856	8672	9 %
4 input LUTs	843	17344	4 %
MULT18X18SIOs	1	28	3 %

FIR filters are most widely used in FPGA implementations because they have linear phase. Compared to IIR filters, FIR filters have simple and regular structures which are easy to implement on hardware. However FIR filters require higher number of taps compared to IIR filters to achieve the same frequency specifications.

References

[1] Vahid, F. – *Digital Design with RTL Design, VHDL and Verilog*, John Wiley & Sons Inc. USA, 2011

[2] Bose, T. – *Digital Signal and Image Processing*, John Wiley & Sons Inc. USA, 2004

[3] Smith, S. W. – *Digital Signal Processing A Practical Guide for Engineers and Scientists*, Newnes, 2003

[4] Oppenheim, A. V. , Schafer, R. W. – *Discrete Time Signal Processing*, Prentice Hall, 1989

[5] Loomis, H.H., Sinha, B. – *High Speed Recursive Digital Filter Realization Circuits*, Systems and Signal Processing, Vol. 3, pp. 267-294, 1984

[6] Meyer – Baese, U. – *Digital Signal Processing with Field Programmable Gate Arrays (Signals and Communication Technology) 3rd Edition*, Springer, 2007