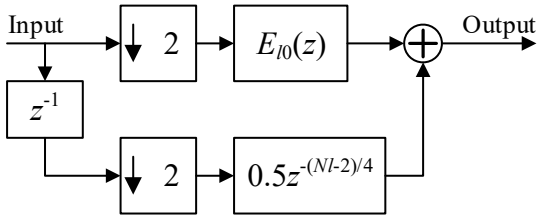

Fig. 2. The l-th stage factor of 2 decimator.

FIR filter of each stage is designed as a half-band filter. However, the required specifications differ from stage to stage. For the decimator considered in this paper, assumed maximal frequency of the signal is f_0, the starting sampling frequency, i.e. sampling frequency before the first decimation stage is f_1 and the resulting sampling frequency i.e. sampling frequency after the decimation is: f_{m+1}. The sampling frequency at the output of the stage l is the same as f_{l+1} the input sampling frequency of the stage l+1:

$$\begin{aligned} f_1 &= 2Mf_0\left(1+\rho\right) \\ f_l &= 2\cdot\frac{M}{2^{l-1}}f_0\left(1+\rho\right),\ 1<l\le m \\ f_{m+1} &= 2f_0\left(1+\rho\right) \end{aligned} \tag{4}$$

where ρ can be considered as a final oversampling factor and for each stage l the output sampling frequency is $f_{l+1}=f_l/2$. The requirements for the normalized (digital) band-edge frequencies for the stage l filters are:

$$\begin{aligned} \omega_{pl} &= 2\pi\frac{f_0}{f_l} = \pi\frac{2^{l-m-1}}{\left(1+\rho\right)} \\ \omega_{sl} &= \pi-\omega_{pl} \end{aligned} \tag{5}$$

where ω_{pl} is the pass-band edge frequency, and ω_{sl} is the stop band edge frequency of the l-th stage half-band filter. It should be noted that the normalized (digital) band-edge frequencies are symmetrical about the π.

From the equations (4) and (5) it is obvious that the transition zone of the first stage filter is the widest comparing to other filters. For that reason, the required order N_l of the l-th stage filter increase from stage to stage, i.e.:

$$N_1\le N_l\le N_m. \tag{6}$$

The overall system is equivalent with the single stage decimator of the factor M, Fig. 3, with the transfer function of the equivalent filter:

$$H_{eq}\left(z\right)=\prod_{l=1}^{L}H_l\left(z^l\right). \tag{7}$$

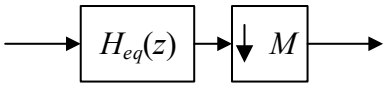

Fig. 3. The equivalent factor of M decimator.

III. FPGA Implementation

The implementation of the proposed multistage FIR filter decimators depends on the specific application and target platform of the overall system. Efficient FPGA implementation can be achieved by time-sharing of the chip resources. Therefore, it is possible to perform complex math operations with relatively low number of hardware-built in multipliers and memories available on the chip. The limiting factor is the relationship between maximum working frequency of the system (F_s) and signal sampling frequency (F_{in}), where filtering takes places and which gives maximum number of possible operations over one period of the signal.

The implementation structure will be explained in details for the example decimator of the overall decimation factor M=16. The decimator is realized as a 4 stage structure with stop-band attenuation of 80 dB. The orders of the half-band l-th stage filters are: N_1=6, N_2=10, N_3=18 and N_4=234. The magnitude response of the equivalent filter is presented in Fig. 4.

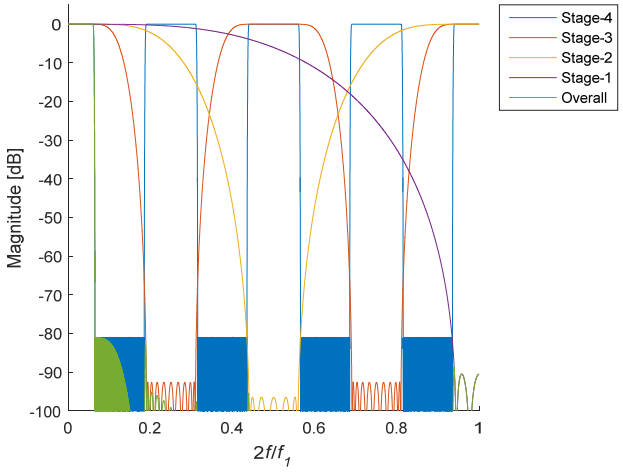

Fig. 4. Magnitude response of the equivalent filter.

This specific half-band filter for decimation is implemented on Spartan 6 slx100 series of Xilinx FPGAs [3]. For the relationship beetwen working frequency and input signal sampling frequency F_s/F_{in} in the system, value 200 is taken. The realization of half-band filter can be divided into the first three stages of decimation, with low number of coefficients and into the fourth stage, which comprises numerous coefficients. During filter implementation, only non-zero coefficients are used, along with filter symmetry, so the number of arithmetic operations is lowered 4 times when compared to direct FIR realization. Coefficients of interest are stored in ROM memory.

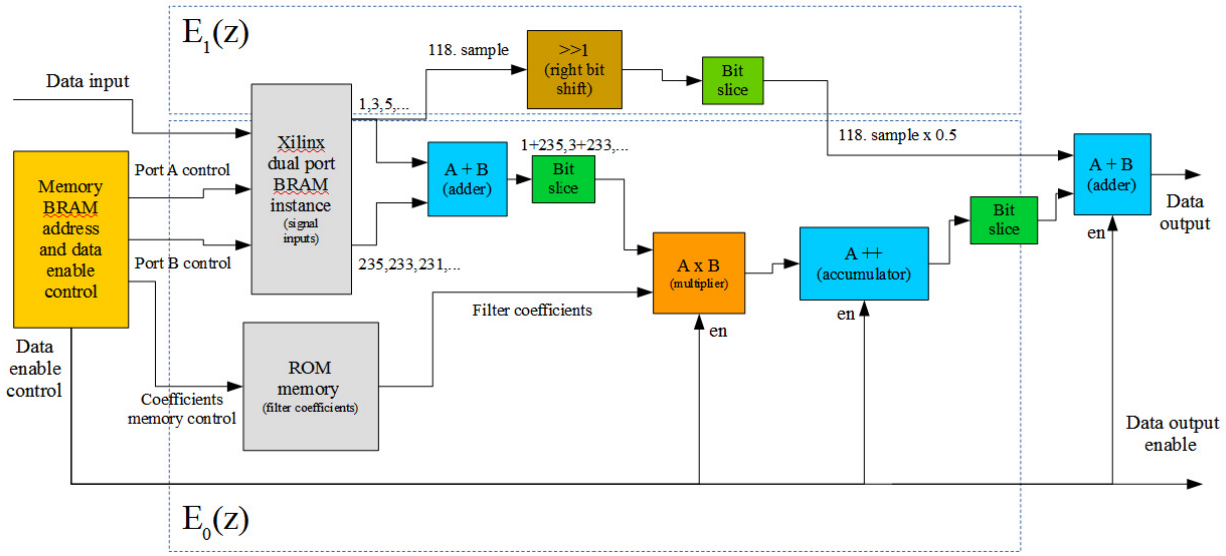

Fig. 5. Efficient FPGA implementation.

Block diagram of implementation is given in Fig. 5. In order to achieve effective filter realization, branches $E_1(z)$ and $E_0(z)$ share mutual Xilinx dual port BRAM where all 235 signal samples are stored. The yellow block in Fig. 5. named "*Memory BRAM address and data enable control*" is in charge of control of data write and read process from the memory, as well as of suitable data enable control signal which is active every second signal sample. On memory port A and memory port B, the first and 235th, the third and 233rd , etc. signal samples are simultaneously read, pair by pair, then they are added and multiplied with corresponding filter coefficients. Multiplier output is taken to accumulator. Trivial filter branch is implemented in such a way that the central signal sample is read from the same memory and by shifting the bit into the right side for one place, the operation of multiplication by 0.5 is achieved.

In the first three stages, when non-zero elements and the symmetry of FIR filters is taken into account, the number of coefficients is in order 3, 5, 9, for these filters, so Distributed RAM (SliceM components on the Spartan 6) can be used for their implementation instead of BRAMs.

This architecture can be used for all filters which fulfill the condition that F_s/F_{in} is larger than the number of non-zero and symmetrically different filter coefficients + 1. Apart from efficiency, the suggested architecture is also modular, i.e. with small changes, it can be used for all levels of decimation, various filter types and working frequencies of FPGAs as well.

The input data type for signal samples is Fix18.17 (word-length is 18 and the number of fractional bits is 17). Inputs for hardware-built in multipliers on Spartan 6 FPGAs are 18-bit wide (Fix18.0), with the result of multiplication operation of 48 bit in length. For the accumulator (i.e. multiplier) output, 48-bit data are used, so slicing of the bit is inevitable at the end of calculation. Furthermore, on adder output, the result is given by the format Fix19.17, so slicing one LSB bit is necessary. In every spot where bit slicing took place, a block "*Bit slice*" is used, which performs the slicing and rounding of the results. By this block, random value is added (0 or 1 for LSB) to the final result, so average value of quantization error is 0 [4]. Block diagram of this block is given in Fig. 6. On the desired length of slicing – Q, from initial data word length of $Q+M$, $M\geq1$, value produced as concatenated zero constant of the length Q-1 and (Q+1)th bit from the initial word, is added. By adding these two values, the desired Q word length is achieved, with zero average value of the quantization error. Fig 7. shows the comparison of results achieved in MATLAB (float point arithmetic) and the results of the simulation of FPGA work (fixed point arithmetic).

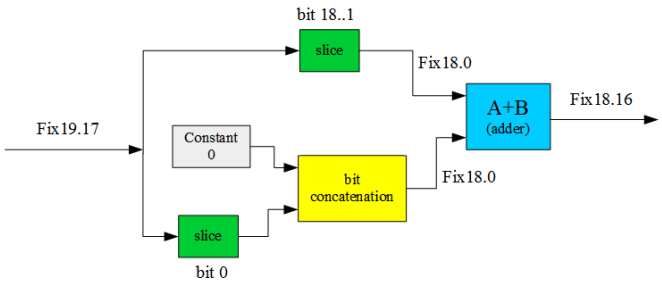

Fig. 6. Rounding by truncation.

TABLE I
FPGA RESOURCES USED FOR HALF BAND FILTER DESIGN ON SPARTAN 6

Component	Used	Available	%
Slices	79	15822	<1%
BRAMs	2	536	<1%
DSPs	1	180	<1%

IV. Round-off Noise Analysis

The results obtained by the single stage filter presented in the previous section are compared with the results obtained in MATLAB with double precision floating point arithmetic. The error of the output signal

$$\Delta out[n] = out_{FPGA}[n] - out_{ML}[n] \tag{8}$$

is shown in Fig. 7. The sources of the error are quantitation of filter coefficients, and quantitation of arithmetic operations. The detailed MATLAB simulation is developed as a tool for investigating the effects of quantization. The simulation approach makes possible investigation of different quantization effects separately. There are three sources of arithmetic operations, and those effects can also be included or excluded form simulation.

The quantization of each source is simulated by means of Monte Carlo simulation, and by means of semi-analytical approach [5]. Monte Carlo simulation requires repetition of the experiment with new set of input signals for each iteration. This approach is time-consuming, however, the implemented structure can be fully simulated. The semi-analytical approach is based on the assumptions about the distributions of error signals introduced by the quantization [6], [7]. Each point of quantization is modeled as an additional source of random signal with defined statistical parameters, amplitude distribution, mean and variation. The influence of the observed noise source at the system output is calculated analytically or numerically by propagating the source signal from its origin to the decimator output. Monte Carlo simulation can be used for adjustment of semi-analytical model.

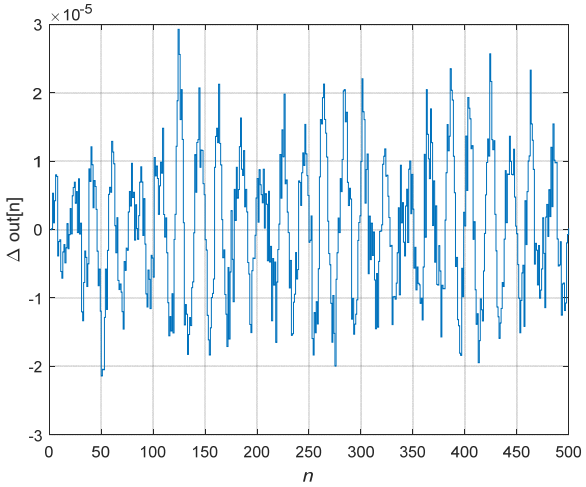

Fig. 7. Output signal error.

As an example, power spectral densities of the noises introduced in each decimator stage as results of the quantization at the output of shifter, multiplier and adder are obtained by means of simulation. It should be noted that the noise introduced in the stage l is low-pass filtered by all filters of the stages l+1,…,L. Because of that, the influence of the last stage is dominant, Fig. 8.

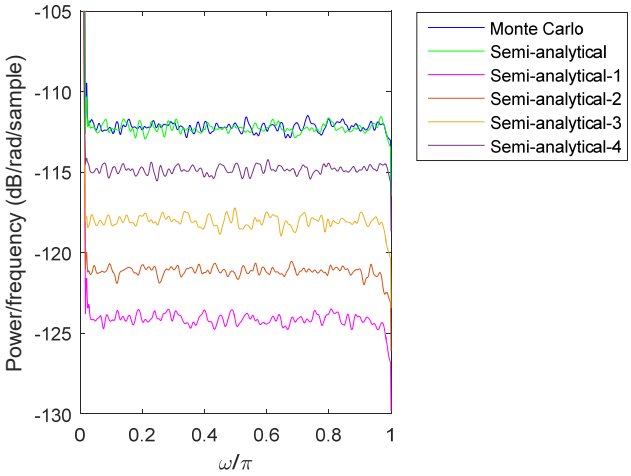

Fig. 8. PSD of the output noise signal.

V. Conclusion

The efficient FPGA implementation of decimator presented in this paper is based on multistage FIR filter design. The implementation is developed in a way that allows it to be change to meet different requirements, for example, different number of stages, or different filter lengths. The results presented in this paper are first step in detailed analysis of the quantization noise propagation in the case of multistage multirate systems.

Acknowledgment

This paper is realized as a part of activities supported by Ministry of Education, Science and Technological Development, project TR-36026 and project TR-32023.

References

[1] L. Milić, *Multirate filtering for digital signal processing: MATLAB applications*, Info. Science Reference, Hershey, PA, 2009.

[2] R. Crochiere and L. Rabiner, "Optimum FIR digital filter implementations for decimation, interpolation, and narrow-band filtering," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 5, pp. 444-456, Oct 1975.

[3] http://www.xilinx.com/support/documentation-navigation/silicon-devices/fpga/spartan-6.html (accessed May, 6th 2017).

[4] C. Maxfield, "An introduction to different rounding algorithms," EETimes, 2006, http://www.eetimes.com/document.asp?doc_id=1274485 (accessed May, 6th 2017).

[5] J. A. Lopez, G. Caffarena, C. Carreras and O. Nieto-Taladriz, "Fast and accurate computation of the roundoff noise of linear time-invariant systems," in *IET Circuits, Devices & Systems*, vol. 2, no. 4, pp. 393-408, Aug. 2008.

[6] A. V. Oppenheim and R. W. Schafer, *Discrete-time signal processing* Prentice Hall, Englewood Cliffs, NJ, 1989.

[7] K. Parashar, *et al.*: 'Shaping probability density function of quantisation noise in fixed point systems', Proc. Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, pp. 1675-1679, Nov. 2010.