IJIEMR Transactions, online available on 1st Aug 2021. Link :http://www.ijiemr.org/downloads.php?vol=Volume-10&issue=ISSUE-08

DOI: 10.48047/IJIEMR/V10/I08/53

Title **A Digital FIR Filter in FPGA Implementation and Performance Comparison**

Volume 10, Issue 08, Pages: 345-350

Paper Authors
Chetti Venkateswarlu, Dr.T.Anil Kumar

A Digital FIR Filter in FPGA Implementation and Performance Comparison

[1]Chetti Venkateswarlu, [2]Dr.T.Anil Kumar
[1]Assistant Professor, Department of Electronics and Communication Engineering,
Malla Reddy Engineering College for Women (Autonomous) Hyderabad
[2] Professor, Department of Electronics and Communication Engineering,
CMR Institute of Technology (Autonomous) Hyderabad
[1]venkatchece@gmail.com [2]tvakumar2000@yahoo.co.in

***Abstract*— There is a discussion of FIR filter design and implementation and performance assessment on a hardware platform in this paper. Digital filters may be synthesized using signed integer terms for the coefficients of filters with limited word length. In this study, the design structure, resource consumption, and occupied silicon space needed for FPGA implementation are all examined. The Gaussian window filter and the Least Square Linear Phase FIR filter are two examples of digital filters that may be evaluated using MATLAB. MATLAB's "HDL coder" tool generates HDL code, which is then used to synthesize and implement the filter designs. The FPGA vendor's pre-implemented Soft-IP Core is also utilized to build a pre-defined FIR filter function. The real-time performance and FPGA resource consumption of the filters are then evaluated on a hardware platform.**

Keywords—Digital filter, FPGA, Gauss window filter, Least square filter, performance comparison

I. Introduction

High sampling rates in comparison to standard DSP chips and reduced costs in comparison to Application Specific Integrated Circuits (ASICs) characterise the FPGA implementation method to digital filtering [1]. (ASIC). Alternative alternatives lack the flexibility that FPGAs provide. Because of the multiple multiply-accumulate (MAC) units in FPGAs, it is possible to implement digital filters with similar performance to general-purpose architectures with a single MAC unit by implementing data path designs encountered in digital filtering applications in a single device using a non-trivial number of arithmetic operations. Different filtering procedures are easy to implement thanks to the FPGA's reprogrammable nature.

The architecture of the digital filter [2] is mostly determined by the target applications to the specific implementations. These filters have highly precise reproducibility that allows the design engineers to achieve the desired performance level which are difficult to obtain with analog filters. Digital filters are broadly classified into Infinite Impulse Response (IIR) digital filter and Finite Impulse Response (FIR) digital filter. Most of the filters algorithms require multiple number of multiplication and addition in real-time. The filter algorithms that require addition and multiplication in real time are carried out by MAC units [3]. In cases where FPGA does not support floating point arithmetic and the desired response must be in linear phase, FIR digital filter is considered as the best candidate [4]. The salient features of FIR filters such as the system stability, high operational speed, linear phase characteristics, design flexibility and low sensitivity to quantization effects made it the most widely used filter in the digital audio, image processing, data transmission, biomedical and other areas with the development of modern electronic technology. FIR filter has achieved a variety of ways, making use of FPGA in digital signal processing technology, FPGA with high speed and reliability, supported the effective implementation of FIR filter.

This paper, discusses the design of FIR filter using two techniques viz. least square linear FIR filter [6] and Gaussian window filter [7] and the implementation of these filters are done by MATLAB generated filter and pre-implemented Soft-IP Core filter. As the targeted FPGA of the hardware platform supports only fixed point arithmetic, the coefficients of the filter must be in the same format. This paper also discusses different approaches to the implementation of digital filter algorithm in FPGA and compares its performances, resource usage and ease of implementation.

The paper is organized as follows. Section II talks about the FIR filter which is the key filter type used for design. Section III describes the methodology where the design flow and the hardware platform are detailed. Section IV gives the comparison of filters and the design details of filters used for the design. The filter implementation into the FPGA hardware platform is described in section V. The performance and resource usage of the filter implementation is discussed in section VI.

II. FIR DIGITAL FILTER

Equations The impulse response of FIR filters has a limited number of samples. The linear combination of the current input and the N prior inputs is the FIR filter's output. As a result, FIR filters are considered non-recursive. Consequently, only a limited amount of non-zero output values are generated from a finite amount of non-zero input values. In FIR filters, the phase and group delays are constant. Using non-recursive approaches, it provides a stable and non-oscillatory implementation. [2] and [4] Nonrecursive implementations may reduce the round-off noise that results from limited precision arithmetic in the digital processor.

The general form of Linear Time Invariant (LTI) FIR system's output at time n is given by

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

where $h[n]$ is the impulse response.

The architecture of Linear Time Invariant FIR filter is graphically shown in Fig.1. FIR filter consists of adders, multipliers and a "tapped delay line". The FIR filter is also known as "transversal filter," because of its "tapped delay line" structure.

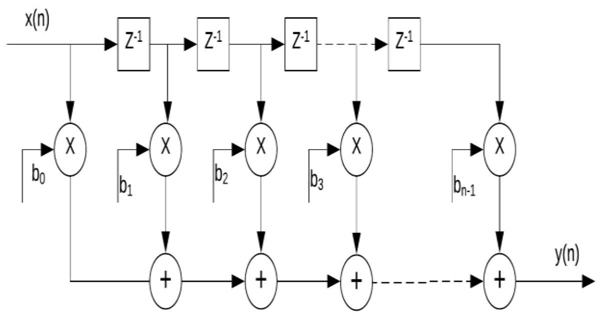

Fig.1. Architecture of FIR filter

The transfer function of a typical FIR filter of filter length M takes the form:

$$H(z) = b_0 + b_1z^{-1} + \cdots + b_{M-1}z^{1-M} = \sum_{n=0}^{M-1} b_n z^{-n} \tag{2}$$

III. METHODOLOGY

A. Design flow

The methodology for implementation of the optimal order FIR filters on FPGA platform is discussed in this section. The work flow of the process is shown below in Fig.2.

The foremost step is to decide the filter specification based on the requirement. The filter is designed using MATLAB [8] and the fixed-point coefficients are derived. The filter model is simulated and tested in MATLAB by using the digital data collected from the hardware. This approach helps in testing the filter with real world signals inclusive of noise associated with hardware platform. Appropriate filter designs are chosen based on this simulation results performed on the real world data.The filters that are chosen based on the different criteria is then implemented in FPGA by two methods that will be detailed in section V. The filters that are implemented in FPGA are compared for their performance and resource usage.

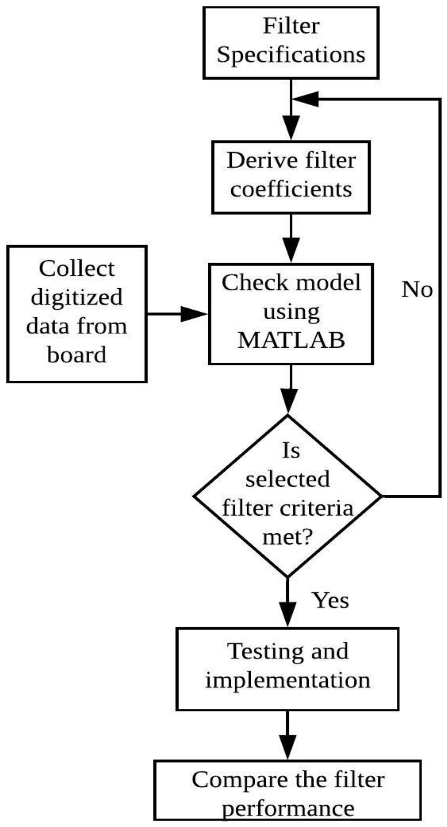
Filter Specifications
Derive filter coefficients
Collect digitized data from board
Check model using MATLAB
No
Is selected filter criteria met?
Yes
Testing and implementation
Compare the filter performance

Fig.2 Design flow of filter design and Implementation

B. Hardware platform

Fig.3 shows the block diagram of hardware platform in which the performance evaluation of filter is carried out. The hardware platform consists of anti-aliasing filter, differential amplifier, Analog to Digital Converter (ADC) and FPGA. The anti-aliasing filter to which the low power signal is fed helps in restricting the bandwidth in order to satisfy the Nyquist sampling criteria over the band of interest. The differential amplifier converts the single ended signal to differential signal and fed to ADC. The analog signal is converted to digital domain using an ADC. A high speed 16-bit ADC used in the hardware which converts this differential output to digitized samples. The hardware supports integrated ADC in order to eliminate the need of external mixed-signal support ICs. The digitized data is then fed to FPGA which have large resource of logic gates and Random Access Memory (RAM) blocks to implement complex digital computations. The FPGA used in this hardware platform consist of logic elements where each logic element is comprised of Look-up-tables (LUT) and Dflip-flops (DFF). It also accommodates approximately 85 math blocks which supports a signed multiplication, dot product and built in addition, subtraction and accumulation units to combine multiplication results efficiently. The size of fabric RAM in FPGA is 2 K bits. The output of the FPGA is saved, verified and compared using the simulation. This hardware platform is a mixed signal board and hence the digitized data of ADC is additive of the source signal and the noise associated with the source and the hardware board.

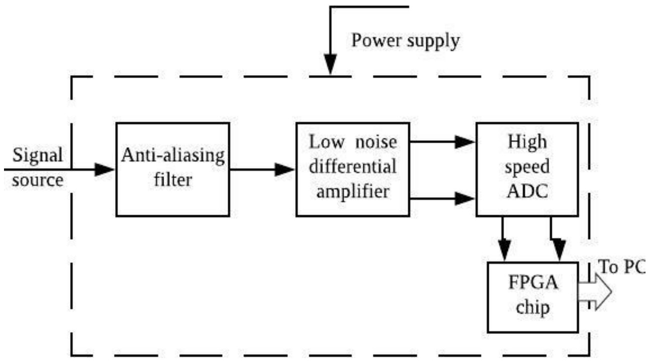

Fig.3 Block diagram of Hardware platform

IV. Design Of Filters

A low pass filter is designed for FPGA based hardware to filter input signal of bandwidth 1MHz to achieve the desired performance. The specification of the filter is decided based on the application and noise floor measured on the hardware platform. The sampling rate of 10 MHz with a cut off frequency of 1.2 MHz, The pass band ripple must be minimum as possible and stop band attenuation below -50 dB. The filter is designed to meet the above requirements. The crucial factor considered while designing the filter is the linear phase response so the selection of filters is confined to the Finite Impulse response filters. While coming to the implementation of filter in FPGA, synthesizable fixed point math is used. This is because of the fact that the targeted FPGA does not support floating-point arithmetic. The coefficients of filters are finite word length, expressed as signed integer by multiplying an integer power of two [9].The number of taps of filter is other factor is taken into consideration while designing efficient filter for the hardware implementation. Also, the quantization factor, which alters the magnitude response of infinite precision floating point math, is considered. Therefore the filter need to be carefully selected based on the variation in both reference and quantized magnitude and phase response. Table I shows the filters that are considered for selection of filters. Based on the results from Table I, the best suited Least square linear phase FIR filter and Gauss window filter are considered for the design and implementation.

A. *Least square FIR filter*

The Least square linear phase FIR filter design is based on Parks-McClellan algorithm which minimizes the weight integrated squared error between desired frequency response and the actual frequency response. The magnitude response of the filter is written as a linear combination of cosines and the mean square error is minimized by solving these equations. The filter is designed to separate the desired signal from undesired signals or noise. An error function is defined by [6].

TABLE.I
COMPARISON OF FILTERS

Filter types		Criteria for selection of filters			
		Linear phase response	***Filter length***	***Response of filter to floating point and quantized coefficients***	***Fixed point math***
FIR	Equiripple filter	Linear	Very high	Vary	Yes
	Least square filter	Linear	low	No variation	Yes
	Gauss window filter	Linear	low	No variation	Yes
	Blackman window filter	Linear	low	Vary	Yes
	Keiser Window	Linear	Very high	Vary	Yes
	Interpolator	Linear	Moder ate	Quantization not applicable	No
IIR	Butterworth filter	Not linear	high	No variation	Yes
	Chebyshev filter	Not linear	low	No variation	Yes
	Elliptic	Not linear	low	No variation	Yes

$$Error(E) = \int_0^{\pi} \mathrm{W}(\mathrm{w})(A(\mathrm{w}) - D(\mathrm{w}))^2 dw \qquad (3)$$

where $A(\mathrm{w})$ and $D(\mathrm{w})$ gives the actual and ideal frequency response respectively, $\mathrm{W}(\mathrm{w})$ is the weighted function which eliminates the Gibbs phenomenon and allows the assignment of different weights to the pass-band and stop band.

B. *Windowed FIR filter*

The window method uses a form of truncating function to achieve a finite duration of output samples from the ideal filter equations. The length of filter can be varied to meet the roll-off rate in the transition band. Low pass filters have been considered in this extract [7].

$$b = \frac{\sin(\mathrm{w}_C[n-M])}{\pi[n-M]} \mathrm{W}_L[n- M] \qquad (4)$$

$$\mathrm{W}_L = \{ \begin{matrix} F_E[n] & 0 \ll n \ll L-1 \\ 0 & otherwise \end{matrix} \qquad (5)$$

where b is the impulse response and W_L is the window function of filter having M number of taps.

The Fourier transform of a Gaussian function is also Gaussian. Since the Gaussian function extends to infinity, it must either be truncated at the ends of the window, or itself windowed with another zero-ended window [4]. The log of a Gaussian produces a parabolic, which is used for exact quadratic interpolation in frequency estimation is given by

$$\mathrm{W}_G(n) = \mathrm{e}^{\frac{-1}{2}\left(\frac{n-N-1)/2}{\frac{\sigma(N-1)}{2}}\right)^2} \qquad (6)$$

where,σ is the standard deviation of the Gaussian function, $-N/2 \leq n \leq N/2$, $\sigma \geq 2$ is the tuning parameter of the window to have the desired "main lobe width – side lobe peak" trade-off and window length is M=N+1. The width of the window is inversely related to the value of σ; a larger value of σ produces a narrower window.

C. Simulation testing

The simulation of designed filter is carried out using MATLAB. The magnitude response of both Least square filter and Gauss window filter are given in fig.4 (a) and (b) and their phase responses are provided in fig.5 (a) and (b). Comparing the magnitude response of both filters the Least square shows a stop band attenuation of -52 dB and Gauss window shows -60 dB. Pass band of least square filter shows some ripples of about 3 dB while the pass band of Gauss window filter is flat without any ripples. The phase responses of both the filters are linear which satisfies the criteria for implementation of filters in to the hardware.

V. Implementation Of Fir Filter

The implementation of FIR filter designed using the filter builder application in MATLAB done by using two methods which are discussed below. The filter is designed by using MATLAB, and then HDL code is generated in order to develop the hardware for MATLAB generated filter. The coefficients generated by the tool are then fed into Soft-IP Core filter for implementation of Soft-IP Core to the hardware. The derived coefficients are initially quantized to 16 bits per coefficient to meet the requirement of a minimum of 50 dB stop band attenuation. Then, the coefficients are quantized converted integer as floating point coefficients are not supported in FPGAs.

A. MATLAB generated filter

The filter designed using the tool is then converted to Hardware Description Language using *hdlcoder* tool. *hdlcoder* generates portable synthesizable Verilog and VHDL code from MATLAB functions, it provides a workflow advisor that automates the programming of FPGAs. The test bench generated by the *hdlcoder* uses the coefficients derived from the designed filter to verify the design by pre synthesis simulation.

B. Soft-IP Core FIR filter

Soft-IP Core filter provides highly configurable area-efficient, high performance FIR filter that utilizes MAC blocks available in the device.

The Core generates the RTL code of filter in Verilog and VHDL. For this FIR filter, the host interface and the FIR filter are implemented in the fabric for low pass, band pass, and band stop filtering operations. The test bench provided for the simulation uses generated filter coefficients, input signals (Pass band frequency and Stop band frequency), and passes the values to the design.

The FIR filter coefficients can be loaded from text file (*.txt). For a symmetric or anti-symmetric filter, only half of the coefficients must be listed in the file. The coefficient of mathematically tested linear phase FIR filter is loaded into filter core.

VI. Results And Discussion

The comparison of the filter performance and resource usage is done for MATLAB generated filters and Soft-IP Core filters by using Least square and Gaussian window filter design methods. The analysis of FFT plot of the signal gives the effective noise reduction of filtered signal compared to the original signal with noise. At the implementation level the efficiency of filter designed is determined by the performance and resource usage of FPGA.

A. FFT analysis

Fig.6 shows the FFT plot of the input and output of filters which are implemented and tested in FPGA. The efficient filter is selected according to its effective reduction in the noise floor of the signal. The comparison shows that the filter effectively works on the board and an effective reduction of noise floor and improved SNR is obtained for each filter. Table.II shows the effective noise floor comparison for various filter implemented in FPGA.

B. Resource usage analysis

Resource usage analysis provides the comparison of different FPGA resources utilized by hardware after the addition of filters. The resource utilization summary is given in Table III. After synthesis, a report of filters shows the usage of the parameters for performance comparison, which decides the hardware complexity and efficiency of the design in terms of low area and more speed.

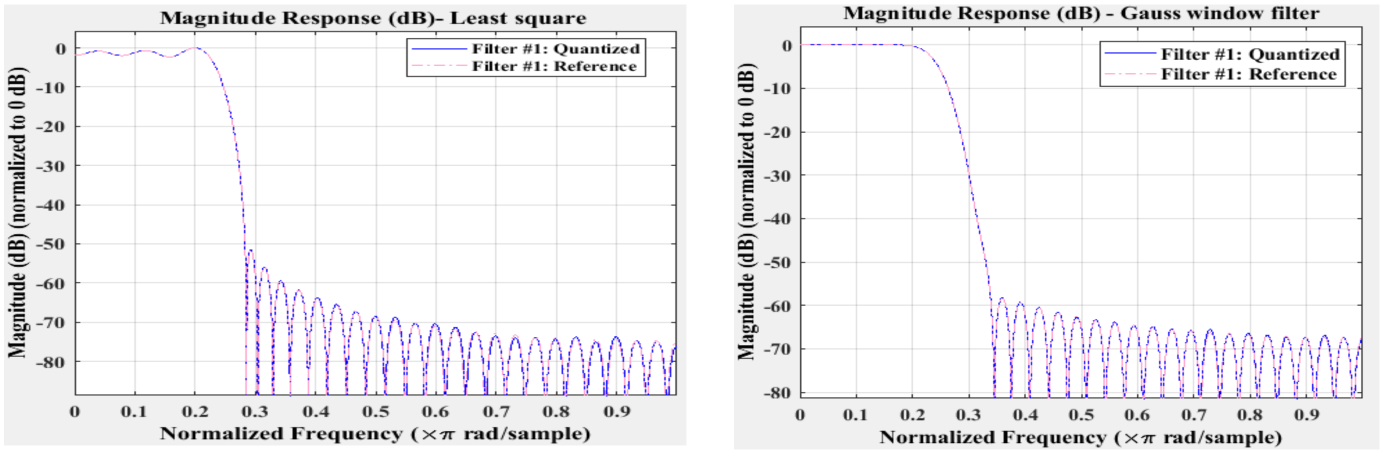

Fig.4 Magnitude response of Least square and Gauss-windowed filter

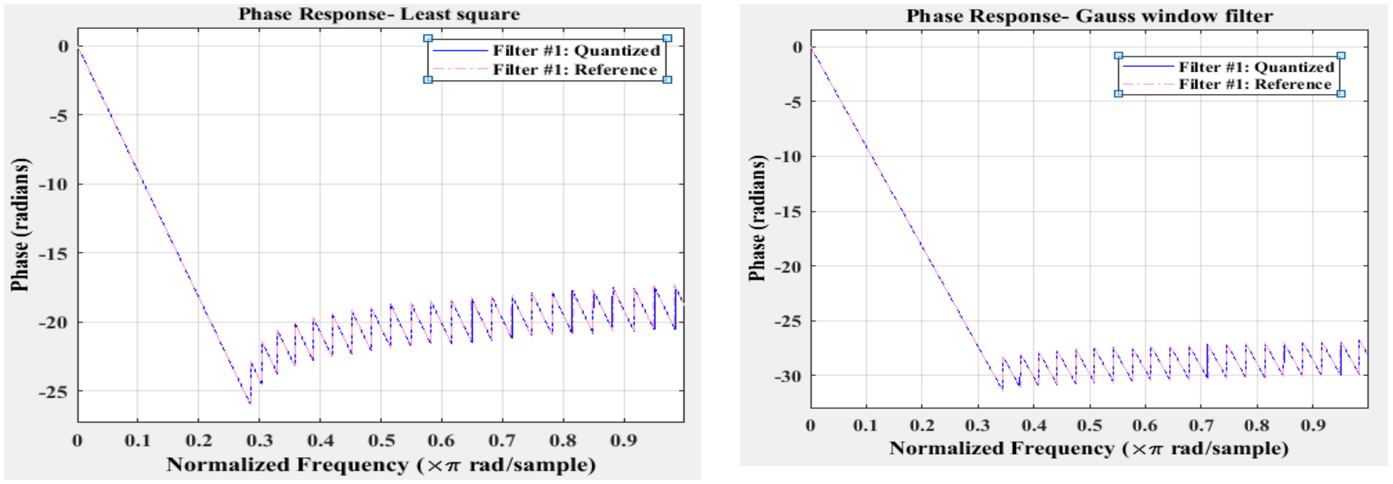

Fig.5. Phase response of Least square and Gauss windowed filter

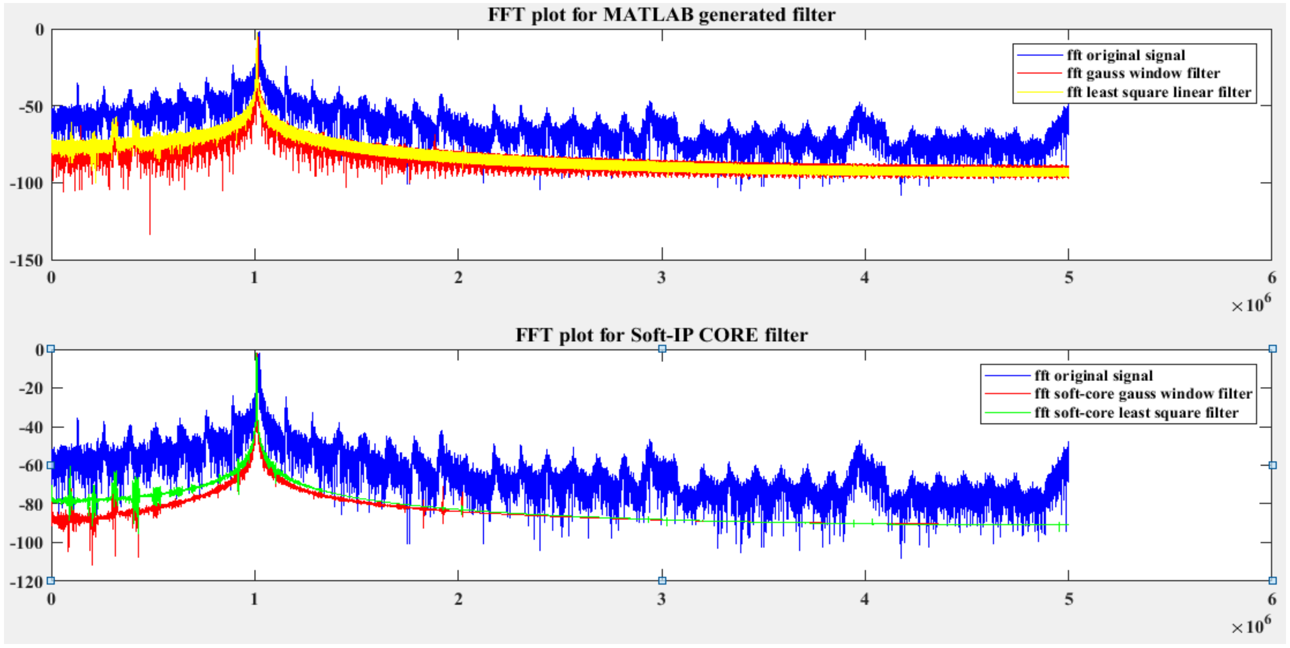

Fig.6.Comparison of filter implementation using FFT plot

TABLE.II
EFFECTIVE NOISE FLOOR COMPARISON FROM FFT PLOT

Effective noise floor	**Input signal (dB)**	**Gauss window filter**		**Least square filter**	
		MATLAB generated (dB)	***Soft-IP Core (dB)***	***MATLAB generated (dB)***	***Soft-IP Core (dB)***
	-60	-88	-90.64	-90.42	-90.9

TABLE.III
RESOURCE UTILIZATION FOR DIFFERENT FILTER USED

Design	***Gauss window***		***Least square***	
Implementation	***MATLAB generated (%)***	***Soft-IP Core (%)***	***MATLAB generated (%)***	***Soft-IP Core (%)***
LUT	7.17	5.93	7.11	5.91
DFF	7.03	7.41	7.18	7.41
RAM Blocks	61.47	70.64	61.47	70.64
MATH blocks	65.48	35.71	70.24	35.71

VII. CONCLUSION

An efficient design and implementation of FIR filters for FPGA applications are the paper's most significant contributions to the field. Filtering undesired high-frequency signals is best achieved using the Gauss window approach and least square error correction, which yields the highest ideal response for pass band ripple and stop band attenuation. For constructing FIR filters, MATLAB is employed since it offers rapid parallel processing. The MATLAB design is used to create HDL code, which is then implemented on an FPGA. In addition, the manufacturer's Soft-IP core is used to implement the filters in FPGA. All solutions are compared in terms of how well they reduce the noise level and how much hardware resources they use. It is recommended that this filter be used with an FPGA that has fixed point math capabilities. Filter types and implementation methods may be chosen based on the performance findings for more effective filtering.

References

[1] Roger Woods,John McAllister, Gaye LightbodyYing Yi “FPGA-based Implementation of Signal Processing Systems”, 2nd Edition, Wiley Publishing ©2017

[2] J. G. Proakis and D. G. Manolakis, “Digital Signal ProcessingPrinciples Algorithms and Applications,” 3rd Edition, Prentice-Hall, Inc., 1996.

[3] V.N Mahadiwar and S.S Shriramwar, “Design of optimized High speed FIR filter” SSRG International journal of Elecxtronics and Communication Engineering 4.1(2017):1-4

[4] B. A. Shenoi, “Introduction to Digital Signal Processing and Filter Design,” John Wiley & Sons, Inc., 2006.

[5] Monmasson, E. &Idkhajine, Lahoucine&Cirstea, M.N. &Bahri, Imen&Tisan, Alin&Naouar, Mohamed. “FPGAs in Industrial Control Applications.IEEE Trans. Industrial Informatics.”(2011). 7. 224-243. 10.1109/TII.2011.2123908..

[6] S. Khorbotly, F. Hassan and R. J. Veillette, "Synthesis of recursive linear-phase filters for fixed-point hardware platforms," in IET Circuits, Devices & Systems, vol. 11, no. 5, pp. 457-464, 9 2017

[7] T. K. Roy and M. Morshed, "Performance analysis of low pass FIR filters design using Kaiser, Gaussian and Tukey window function methods," 2013 2nd International Conference on Advances in Electrical Engineering (ICAEE), Dhaka, 2013,

[8] Ricardo A. Losada “Digital Filters with MATLAB”. The MathWorks, Inc., May 18, 2008.

[9] Uwe Meyer-Baese, “Digital Signal Processing with Field Programmable Gate Arrays”, Third Edition, 2007

[10] Z. G. Feng and K. F. C. Yiu, "Design of prototype filter of DFT filter banks with low implementation complexity," 2009 IEEE 13th International Symposium on Consumer Electronics, Kyoto, 2009,

[11] Volnei A. Pedroni “Circuit Design with VHDL”. MIT Press Cambridge, Massachusetts London, England, 2004.