

Design of FIR Filter for Audio Application

Suraj R. Gaikwad^{#1}, Snehal R. Gaikwad^{#2}, Kunal N. Dekate^{#3}

#1 D.M.I.E.T.R., Wardha, +917755904393.

#2 Arvi, Dist. Wardha, +919561650367.

#3 G.H.R.C.O.E. Nagpur, +917755904391.

ABSTRACT

This paper presents the implementation of FIR (Finite Impulse Response) filter using Xilinx system generator and Matlab Simulink for audio application. This Structure uses noisy audio signal with 8 KHz frequency and choose pass band frequency of 4 KHz. This is a singlerate structure for FIR filter. Main aim of this design is to reduce the noise from given audio signal. This is the 13 tap (N=13) filter structure which having stop band frequency is 7 KHz with 1 dB pass band attenuation and 80 dB stop band attenuation.

Keywords: FIR (Finite Impulse Response) filter, MAC based FIR filter, Impulse response, Transition Band, System Generator.

INTRODUCTION

FIR filters are one of two primary types of digital filters used in Digital Signal Processing (DSP) applications. "FIR" means "Finite Impulse Response". If you put in an impulse, that is, a single "1" sample followed by many "0" samples, zeroes will come out after the "1" sample has made its way through the delay line of the filter. In the common case, the impulse response is finite because there is no feedback in the FIR [1]. A lack of feedback guarantees that the impulse response will be finite. Therefore, the term "finite impulse response" is nearly synonymous with "no feedback".

However, if feedback is employed yet the impulse response is finite, the filter still is a FIR. An example is the moving average filter, in which the Nth prior sample is subtracted (fed back) each time a new sample comes in. This filter has a finite impulse response even though it uses feedback: after N samples of an impulse, the output will always be zero [5].

TERMS ARE USED IN FIR FILTERS

Impulse Response - The "impulse response" of a FIR filter is actually just the set of FIR coefficients [6]. (If you put an "impulse" into a FIR filter which consists of a "1" sample followed by many "0" samples, the output of the filter will be the set of coefficients, as the 1 sample moves past each coefficient in turn to form the output.)

Tap - A FIR "tap" is simply a coefficient/delay pair. The number of FIR taps, (often designated as "N") is an indication of 1) the amount of memory required to implement the filter, 2) the number of calculations required, and 3) the amount of "filtering" the filter can do; in effect, more taps means more stop band attenuation, less ripple, narrower filters, etc [6].

Multiply-Accumulate (MAC) - In a FIR context, a "MAC" is the operation of multiplying a coefficient by the corresponding delayed data sample and accumulating the result. FIRs usually require one MAC per tap [1]. Most DSP microprocessors implement the MAC operation in a single instruction cycle.

Transition Band - The band of frequencies between passband and stopband edges is called as transition band. The narrower the transition band, the more taps are required to implement the filter [2]. (A "small" transition band results in a "sharp" filter.)

Delay Line - The set of memory elements that implement the " Z^{-1} " delay elements of the FIR calculation.

IMPLEMENTATION OF FIR FILTER

The implementation of FIR filter structure is done in Xilinx System Generator 9.1i and Matlab Simulink 2006b. This is 13 Tap FIR filter structure with the pass band frequency is 4 KHz. This design uses FIR compiler from system generator toolbox.

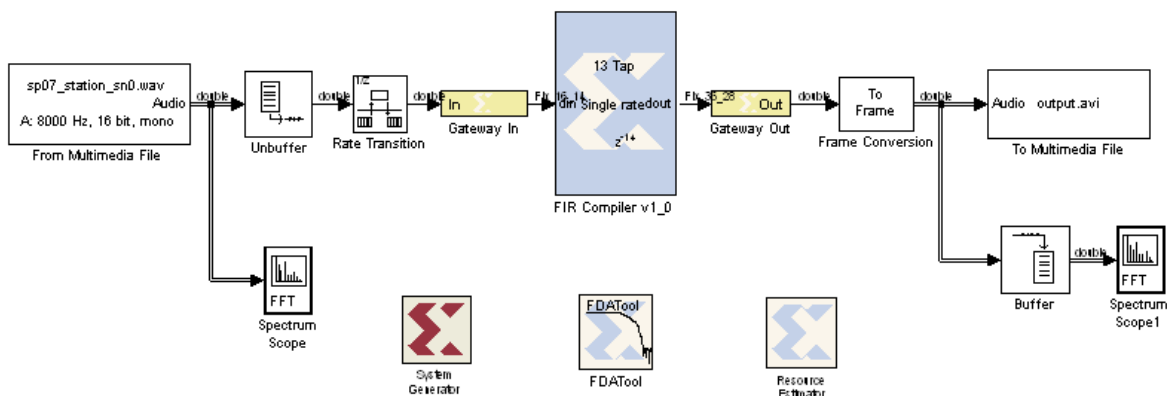


Fig 1: Implementation of 13 Tap FIR filter structure

The From Multimedia File block reads audio samples, video frames, or both from a multimedia file. The block imports data from the file into a Simulink model [2]. We select here the audio signal with frequency 8000 Hz, 16 Bit, mono. The “Unbuffer” block unbuffers an M_i -by- N input into a 1-by- N output. That is, inputs are unbuffered *row-wise* so that each matrix row becomes an independent time-sample in the output. The rate at which the block receives inputs is generally less than the rate at which the block produces outputs [2].

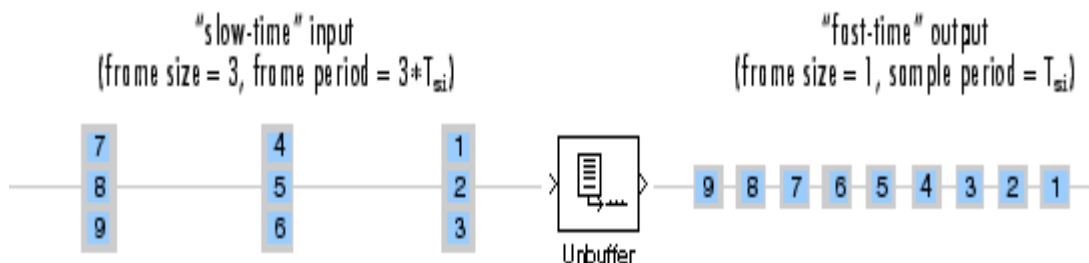


Fig 2: Operation of Unbuffer block

The Buffer block always performs frame-based processing. The block redistributes the data in each column of the input to produce an output with a different frame size. Buffering a signal to a larger frame size yields an output with a *slower* frame rate than the input [2]. For example, consider the following illustration for scalar input.

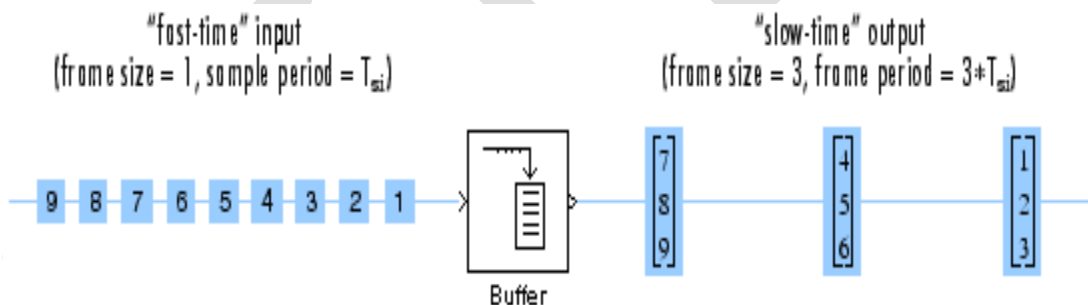


Fig 3: Operation of Buffer block

The Rate Transition block transfers data from the output of a block operating at one rate to the input of a block operating at a different rate. Use the block parameters to trade data integrity and deterministic transfer for faster response or lower memory requirements.

The Xilinx Gateway In blocks are the inputs into the Xilinx portion of Simulink design. These blocks convert Simulink integer, double and fixed-point data types into the System Generator fixed-point type [3]. Each block defines a top-level input port in the HDL design generated by System Generator. The Xilinx Fir Compiler v1_0 block implements a high speed MAC-based FIR filter. It accepts a stream of input data and computes filtered output with a fixed delay, based on the filter configuration. The filter is implemented using cascaded DSP48 slices as shown in the figure below [2].

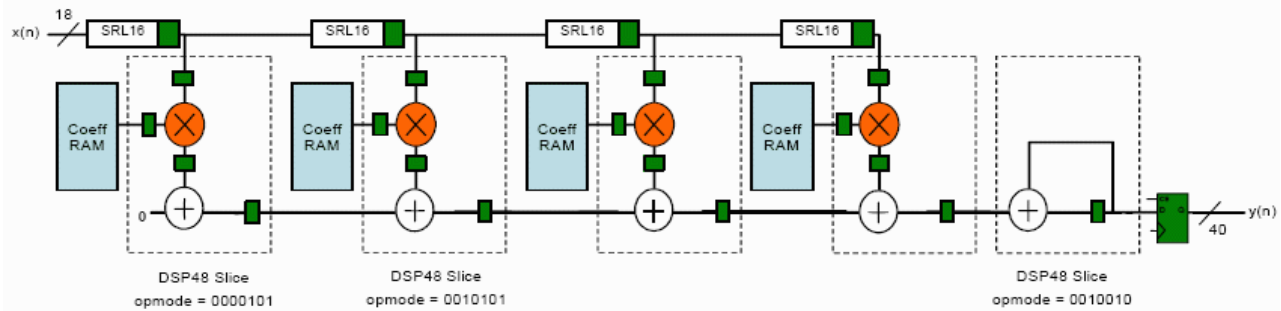


Fig 4: Internal structure of Xilinx FIR Compiler v1_0 block using high speed MAC-based FIR filter

Xilinx Gateway Out blocks are the outputs from the Xilinx portion of your Simulink design. This block converts the System Generator fixed-point data type into Simulink Double [1]. The Frame Conversion block passes the input through to the output and sets the output sampling mode to the value of the Sampling mode of output signal parameter, which can be either Frame based or Sample-based. The “To Multimedia File block” writes video frames, audio samples, or both to a multimedia (.avi, .wav, .wma, .mp4, .ogg, .flac, or .wmv) file [2].

The System Generator block provides control of system and simulation parameters, and is used to invoke the code generator [4]. Every Simulink model containing any element from the Xilinx Blockset must contain at least one System Generator block. Once a System Generator block is added to a model, it is possible to specify how code generation and simulation should be handled.

The Xilinx FDA Tool block provides an interface to the FDA Tool software available as part of the MATLAB Signal Processing Toolbox. The block does not function properly and should not be used if the Signal Processing Toolbox is not installed. This block provides a means of defining an FDA Tool object and storing it as part of a System Generator model [4]. FDA Tool provides a powerful means for defining digital filters with a graphical user interface.

The Xilinx Resource Estimator block provides fast estimates of FPGA resources required to implement a System Generator subsystem or model. These estimates are computed by invoking block-specific estimators for Xilinx blocks, and summing these values to obtain aggregated estimates of lookup tables (LUTs), flip-flops (FFs), block memories (BRAM), 18x18 multipliers, tristate buffers, and I/Os.

Sampling Frequency	16 KHz
Pass Band Frequency	4 KHz
Stop Band Frequency	7 KHz
Pass Band Attenuation	1 dB
Stop Band Attenuation	80 dB
Density Factor	16
Filter order	12

Table 1: Filter Specification

Filter Structure	Direct-Form FIR
Filter Length	13
Stable	Yes
Linear Phase	Yes (Type -1)
No. of Multiplier	13
No. of Adders	12
No. of States	12
Multiplier Per Input Samples	13
Adder Per Input Sample	12

Table 2: Filter Information

RESULT AND SIMULATION

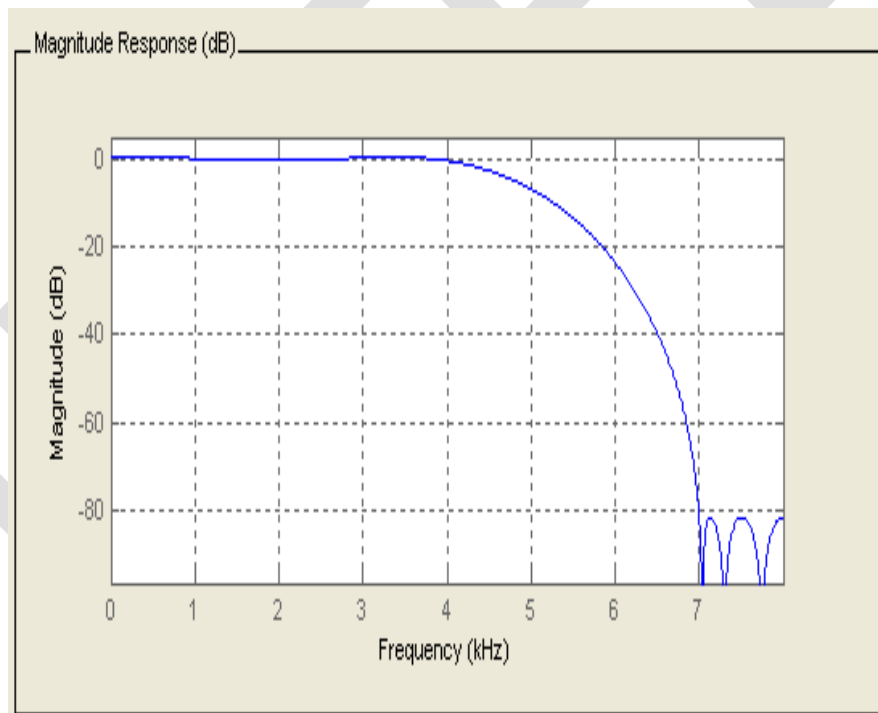


Fig 5: Magnitude response of FIR filter

Above figure shows that the magnitude response of implemented FIR filter structure for filter order is 12. This shows that the stop band attenuation is 80 dB, Pass band frequency is 4 KHz and stop band frequency is 7 KHz.

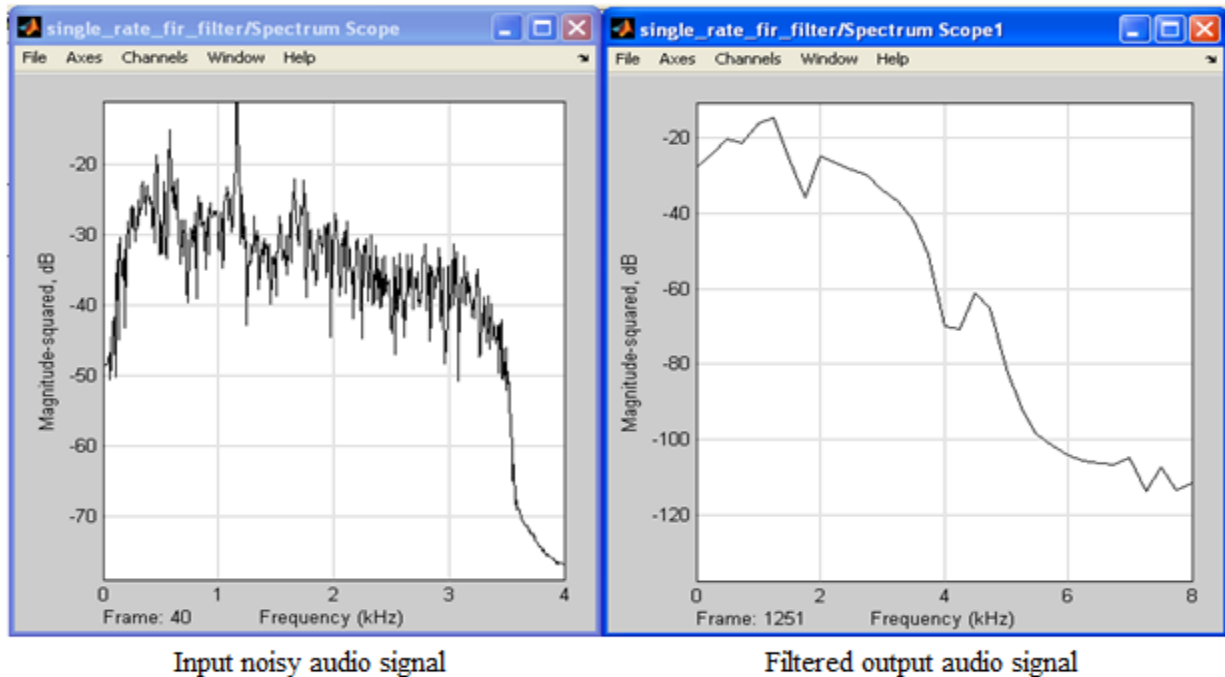


Fig 6: Comparison of Input noisy audio signal and filtered output audio signal

Above figure shows that the comparison between input noisy audio signal of and filtered output audio signal using the FIR filter which filtered the 8 KHz frequency signal and produce the 4 KHz frequency signal.

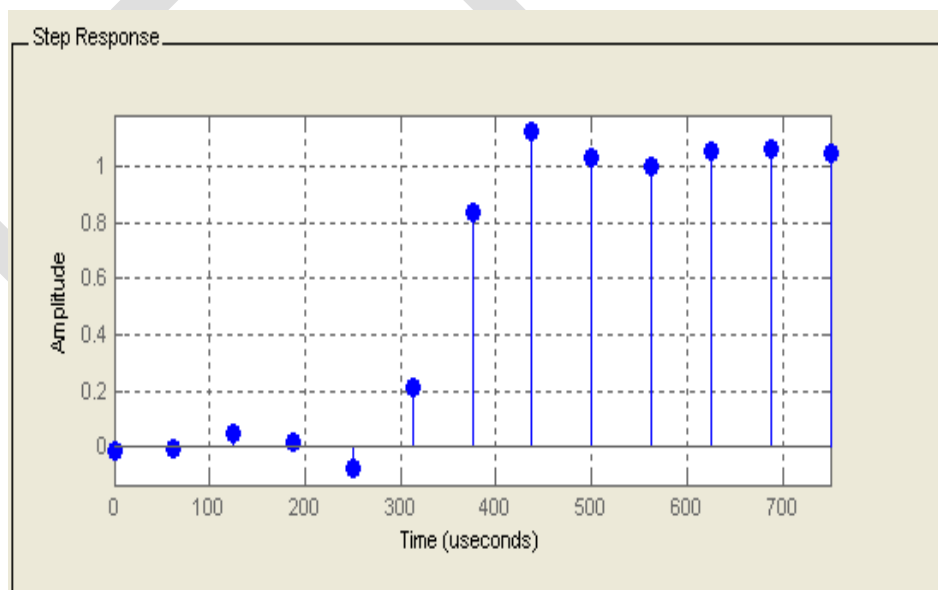


Fig 7: Step Response of FIR filter

This figure shows that the step response of FIR filter which having filter length is 13. Thus the total numbers of coefficients are 13 and hence the step response is plotted in between amplitude and time to representing the filter coefficients.

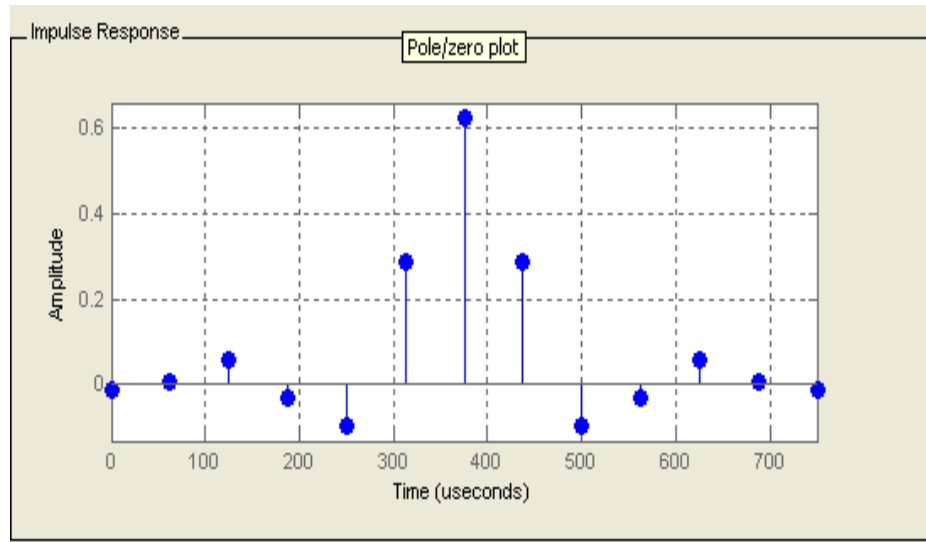


Fig 8: Impulse Response of FIR filter

This figure shows that the impulse response of FIR filter which having filter length is 13. The impulse response is plotted in between amplitude and time to representing the filter coefficients. The impulse response of a FIR filter is actually just the set of FIR coefficients.

CONCLUSION

The implementation of FIR filter structure for audio application is designed with filter length of 13. Thus the requirement of the multiplier is 13 with number of filter coefficients is 13. This design uses number of adders is 12. Thus to reduce the significant noise from the given audio signal, 13 tap FIR filter is efficient in case of hardware efficient structure requirement.

REFERENCE

- [1] Ricardo A. Losada, Digital Filters with MATLAB, *The MathWorks, Inc.* May 18, 2008
- [2] System Generator for DSP, *Reference Guide*, March, 2008
- [3] Manoj Barnela, Suresh Kumar, Akhil Kaushik, Satvika, Implementation and Performance Estimation of FIR Digital Filters using MATLAB Simulink, *International Journal of Engineering and Advanced Technology (IJEAT)*, Volume-3, Issue-5, June 2014
- [4] Suraj R. Gaikwad, Gopal S. Gawande, Design and Implementation of Efficient FIR Filter Structures using Xilinx System Generator, *International Journal of scientific research and management (IJSRM)*, Volume-2, 2014
- [5] Emmanuel Ifeachor and Barrie W. Jervis, Digital Signal Processing – A practical approach, *Pearson Education, Second edition*, 2014
- [6] Jonh G. Proakis and Dimitris G. Manolakis, Digital Signal Processing – Principlas, Algorithms and Applications, *Pearson Education, Fourth edition*, 2011

[7] M. Lang, Allpass filter design and applications, *IEEE Transactions on Signal Processing*, vol. 46, pp. 2505–2514, September 1998

[8] M. Lutovac, D. Tasic, and B. Evans, Filter Design for Signal Processing Using MATLAB and Mathematica, *Upper Saddle River, New Jersey: Prentice Hall*, 2001

RECEIVED