Implementation of Pipelined IIR Bandstop Filter

Abhinav Moudgil

Lincy Pattanaik

***Abstract*—In the context of Digital Signal Processing, the goal of bandstop filtering is to perform frequency dependent alteration of signal that allows low and high frequency components and attenuates the frequencies in the middle band. Here we implement a pipelined IIR bandstop filter following butterworth characteristics in Verilog. The design is simulated in MATLAB and detailed analysis of RTL, placement routing, timing is done in Xilinx. The filter is also implemented in C++ with 32 bit fixed point arithmetic using the filter coefficients obtained from MATLAB. Additional testing of the design is done on Altera Cyclone II FPGA with audio codec interface.**

***Keywords*—bandstop, IIR filter, altera cylone II, xilinx, fixed point arithmetic**

I. Introduction

In signal processing, the function of a filter is to remove unwanted parts of the signal, such as random noise, or to extract useful parts of the signal, such as the components lying within a certain frequency range. An ideal filter is a network that allows signals of only certain frequencies to pass while blocking all others. The goal of bandstop filtering is to perform frequency dependent alteration of signal that allows low and high frequency components and attenuates the frequencies in the middle band. Bandstop filters can be broadly classified into two categories- FIR and IIR. IIR stands for Infinite Impulse Response since the impulse response in this kind of filter is infinite. These are recursive in nature and use feedback mechanism unlike FIR filters. Fig 1 shows the block diagram for a general n-order IIR filter. The general difference equation for IIR filter can be written as

$$\sum_{l=0}^{p} a_l x[n-l] = \sum_{k=0}^{q} b_k x[n-k]$$

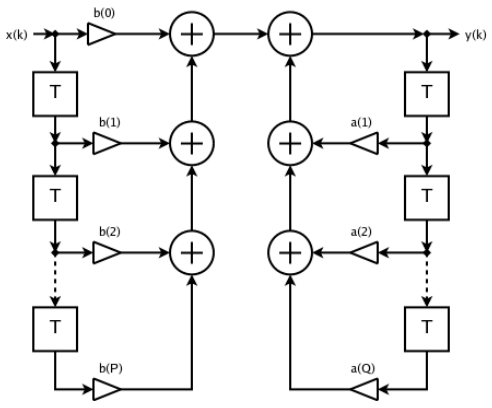

Fig. 1. Block diagram for P+Q order IIR filter

II. IIR filter design

IIR bandstop filter of following specifications

TABLE I. Specifications

Order	12
Sampling Frequency	48 kHz
Stopband Edges	5 kHz, 7 kHz

is designed following butterworth characteristics. The coefficients are calculated using `butter` function in MATLAB.

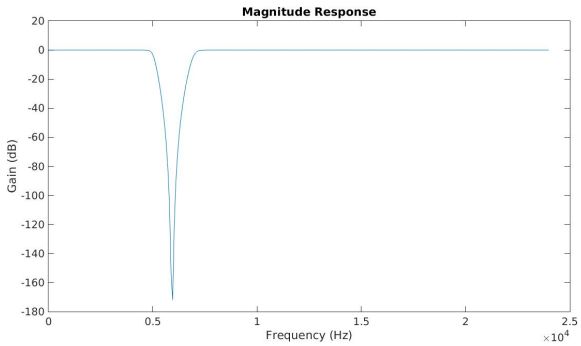

Fig. 2. Magnitude response of the IIR filter

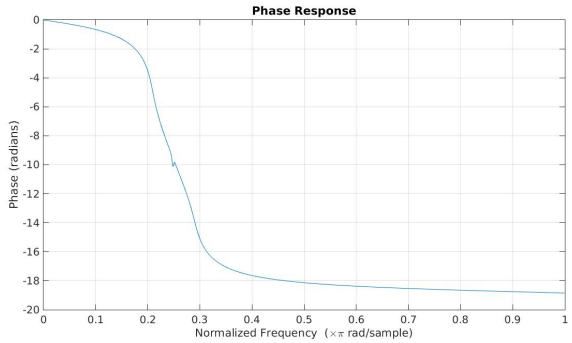

Fig. 3. Phase response of the IIR filter

The filter is then refined to pipelined design.

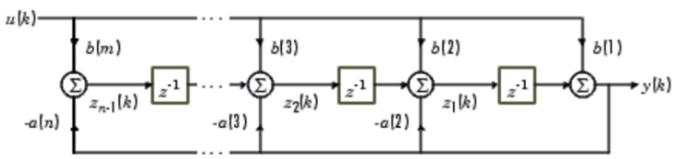

Fig. 4. Pipelined IIR filter

III. Fixed Point Arithmetic

The coefficients obtained from MATLAB are floating point point numbers. These floating point numbers are mapped to integer values to realise the digital IIR filter. Mapping of floats to integers and vice versa is done by Q factor scaling. Q factor is chosen to be 2^{20}. To convert a number from floating point to Qm.n format, we first multiply the floating point number by Q(2^{20}) and then round to the nearest integer. Similarly to convert a number from Qm.n format to floating point, first convert the number to floating point as if it were an integer, in other words remove the binary point and then multiply by2^{-20}.

A. C++ implementation

The IIR filter is also implemented in C++ using the coefficients obtained from MATLAB in 32 bit fixed point arithmetic. 20 bits are kept for fraction, 11 bits before decimal and 1 bit for sign.

TABLE II. Mapping coefficients from float to integer

Coefficient	Floating Value	Integer Value
a1	1	1048576
a2	-7.83747622306246	-8218189
a3	30.6201413194013	32107544
a4	-77.4548985375249	-81217352
a5	140.263174370713	147076592
a6	-190.725578269939	-199990256
a7	199.258637352792	208937824
a8	-161.032629544213	-168854944
a9	99.9871743564717	104844152
a10	-46.6155561037670	-48879952
a11	15.5583749219295	16314139
a12	-3.36225921691592	-3525584
a13	0.362330967823191	379931

TABLE III. Mapping coefficients from float to integer

Coefficient	Floating Value	Integer Value
b1	0.601939338980226	631178
b2	-5.15169815333243	-5401947
b3	21.9828074462778	23050644
b4	-60.6984197266264	-63646908
b5	119.892950811457	125716872
b6	-177.664081067276	-186294288
b7	202.094438094748	211911376
b8	-177.664081067276	-186294288
b9	119.892950811457	125716872
b10	-60.6984197266264	-63646908
b11	21.9828074462778	23050644
b12	-5.15169815333243	-5401947
b13	0.601939338980226	631178

IV. Verilog Synthesis

Using the integer coefficients obtained from C++ implementation, the pipelined version of IIR filter is written in Verilog and tested on 2 testbenches, one with input signal frequency 2 kHz and another with input signal frequency 6 kHz. According to bandstop filter specifications, the output for 2 kHz signal is nearly same as that of input (Fig 5) but 6 kHz signal is blocked (Fig 6). The results matches exactly with the C++ implementation of the same.

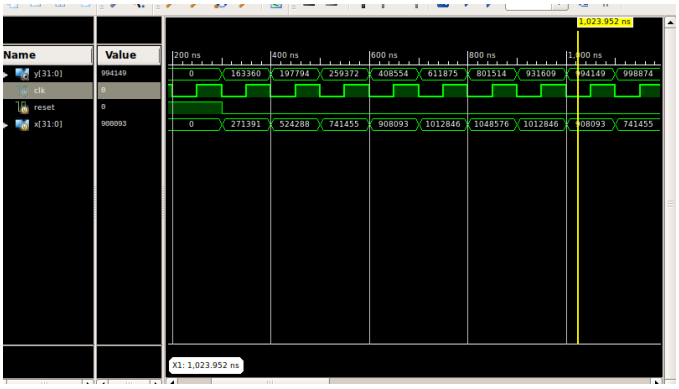

Fig. 5. Test results for 2 kHz input signal

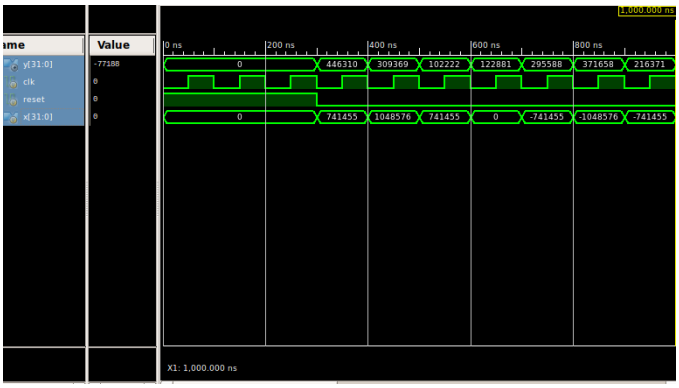

Fig. 6. Test results for 6 kHz input signal

V. FPGA Audio Codec Implementation

The Verilog code is then dumped on Altera Cyclone II FPGA board. Using audio codec, the filter is tested using audio signals below the stopband frequency and in between the stopband frequencies.

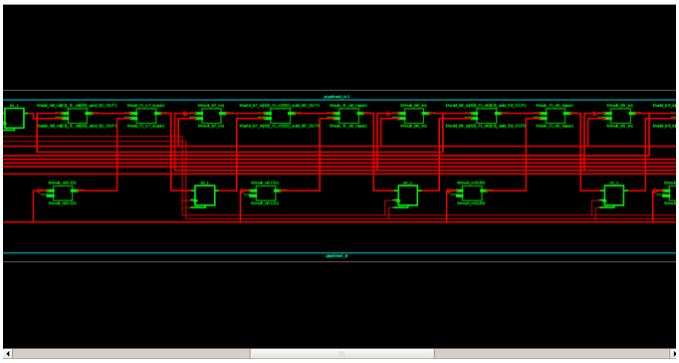

Fig. 7. A snapshot of RTL

References

[1] Chakraborty, Subhadeep. "Design and Realization of IIR Digital Band Stop Filter Using Modified Analog to Digital Mapping Technique." *International Journal of Science, Engineering and Technology Research*, 2.3 (2013): pp-742. [pdf].

[2] Linear Digital Filtering I [pdf].