FIR Filter Implementation on A FPGA Allowing Signed and Fraction Coefficients with Coefficients Obtained Using Remez Exchange Algorithm

Animesh Panda* animesh_6988@yahoo.co.in

Satish Kumar Baghmar* baghmar.satish@gmail.com

Shailesh Kumar Agrawal* shailesh.lormi@gmail.com

T.Siva Kumar* siva.kumar.1678@gmail.com

T. Usha usha156@gmail.com
Department of Mathematics, BIT Durg, India

*Department of Electronics & Telecommunication BIT DURG , India491001

Abstract

A filter may be required to have a given frequency response, or a specific response to an impulse, step, or ramp, or simulate an analog system. Depending on the response of the system, digital filters can be classified into Finite Impulse Response (FIR) filters & Infinite Impulse Response (IIR) filters. FIR Filters can be designed using frequency sampling or windowing methods. But these methods have a problem in precise control of the critical frequencies. In the optimal design method, the weighted approximation error between the actual frequency response and the desired filter response is spread across the pass-band and the stop-band and the maximum error is minimized, resulting in the pass-band and the stop-band having ripples. The peak error can be computed using a computer-aided iterative procedure, known as the Remez Exchange Algorithm.

Key words: Fir filter, Generic, FPGA

1. Introduction

The paper briefly introduces the general aspects of FIR filter. Structure and functions are also discussed. Following the general view we will describe a generic FIR filter written in behavioral VHDL Code. This defines the requirements on a filter generation program in VHDL and its implementation on FPGA. We will also discuss the calculation of coefficients of filter using MATLAB.

2. FIR Filter

FIR filters are a special kind of digital filters. They are non-recursive type of filter where the present output depends on the present input sample and the previous samples. The impulse response of FIR filter has finite number of non-zero terms. Few characteristics of FIR which serve as their advantage are enlisted below:

i. FIR filters have exactly linear phase.

ii. FIR filters are always stable.
iii. FIR filters may be realized in both recursive and non- recursive structures.
iv. FIR filters with any arbitrary magnitude response can be tackled using FIR sequence.

2.1 Structure of FIR filter

The structure of a general FIR filter is shown in fig 1.

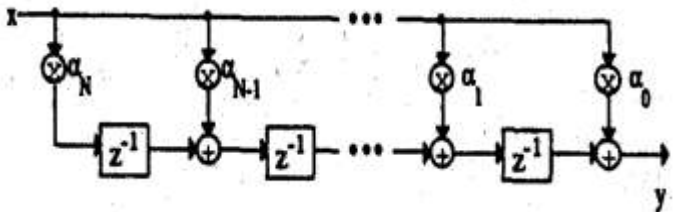

Fig 1: FIR filter structure

From this structure the transfer function of the filter can be easily described in z- domain as:

$$H(z) = \alpha_0 + \alpha_1.z^{-1} + \alpha 2.z^{-2} + \ldots\ldots\ldots + \alpha_N.z^{-N}$$

Where N: filter order

Alternatively,

$$H(Z) = \frac{Y(Z)}{X(Z)} = \sum^{M} b_k . Z^{-k} . X(Z) \qquad \ldots\ldots\ldots (1)$$

The task of designing such a filter is therefore the determination of the filter coefficients α_i. Two common methods applied to solve this problem are:

1. Windowing method, and
2. Iterative method.

Windowing method is much easier to implement. It is a straightforward approach. If we assume that $H_d(e^{jw})$ to be ideal response and $h_d(n)$ to be corresponding infinite-duration impulse response sequence, then we can obtain a finite-duration causal impulse response by multiplying $h_d(n)$ with finite-duration " window" w(n)

$$h_{finite}(n) = h_d(n) . w(n)$$

Window functions for filter design should taper smoothly to a value of zero at each end. Two commonly used windows are the Hamming and the Blackman window. These windows are defined by the following equations:

$$\text{Hamming: } w(n) = 0.54 - 0.46 \cos(2.\pi.n / N)$$
$$\text{Blackman: } w(n) = 0.42 - 0.5 \cos(2.\pi.n / N) + 0.08\cos(4.\pi.n / N)$$

where n= 0,1, …….N

The iterative method allows the design of optimal filters. Optimal filter means filter with constant equiripple in the pass band and the stop band. The most commonly applied algorithm used is the Remez Exchange Algorithm. This algorithm is being discussed below.

3. Algorithm Used

The optimal filter uses the iterative method known as Remez Exchange Algorithm. This algorithm is also known by the names: Remez Algorithm, Remes Algorithm, Remes Exchange Algorithm or simply Exchange Algorithm. It was published by Evgeny Yakovlevich Remez in 1934. It is an iterative algorithm used to find simple approximations to functions, specifically, approximations by functions in the Chebyshev space.

3.1 Procedure

The Remez algorithm starts with the function f to be approximated and a set X of $n + 2$ sample points x_1, $x2$, in the approximation interval, usually the Chebyshev nodes linearly mapped to the interval. The steps are:

1. Solve the linear system of equations
   $b_0 + b_1 x_i ... b_n x_i^n + (-1)^i E = f(x_i)$ (For $i = 1, 2, n + 2$)
   For the unknowns b_0, $b_1 ... b_n$ and E.
2. Use the b_i as coefficients to form a polynomial P_n.
3. Find the set M of points of local maximum error $| P_n(x) - f(x) |$.
4. If the errors at every $m \in M$ are of equal magnitude and alternate in sign, then P_n is the minimax approximation polynomial. If not, replace X with M and repeat the steps above.

The result is called the polynomial of best approximation, Chebyshev approximation, or the minimax approximation.

4. Scaling of the Coefficients

The implementation of the filter uses a fixed point method to represent data. But the coefficients used in the design of the filter is going to be fractional in nature and also it may be negative. So to represent fractions the method of scaling of data has been used. The scaling was done using scaling factors of 2's complement and then shifting the data to the left or to the right. For example if 0.375 is one coefficient which is to be multiplied by 2 to get 0.75 as the answer, then first 0.375 is represented in digital as 0.0110. For doing the above operation first the data is shifted to the left by four bits giving 0110.0, this data is then multiplied by 2 which gives 1100.0. Now the data 1100.0 is again shifted to the right by four bits which gives 0.1100 this value is the same as the required answer of 0.75. An extra bit has been has been included for considering the overflow of the data.

5. Implementation

The implementation has been explained below starting with calculation of coefficients using MATLAB, followed by VHDL program and its implementation:

5.1 Coefficient calculation in MATLAB

The Remez FIR Filter Design block implements the *Parks-McClellan algorithm* to design and apply a linear-phase filter with an arbitrary multiband magnitude response. The filter design, which uses the *firpm* function in Signal Processing Toolbox, minimizes the maximum error between the desired frequency response and the actual frequency response. Such filters are called equiripple due to the equiripple behavior of their approximation error. The block applies the filter to a discrete-time input using the *Direct-Form II* Transpose Filter block.

An M-by-N sample-based matrix input is treated as M*N independent channels and an M-by-N frame-based matrix input is treated as N independent channels. In both cases, the block filters each channel independently over time, and the output has the same size and frame status as the input.

The Filter type parameter allows you to specify one of the following filters:

- ***Multiband:*** The multiband filter has an arbitrary magnitude response and linear phase.
- ***Differentiator:*** The differentiator filter approximates the ideal differentiator. Differentiators are antisymmetric FIR filters with approximately linear magnitude responses. To obtain the correct derivative, scale the Gains at these frequencies vector by pFs rad/s, where Fs is the sample frequency in Hertz.
- ***Hilbert Transformer:*** The Hilbert transformer filter approximates the ideal Hilbert transformer. Hilbert transformers are antisymmetric FIR filters with approximately constant magnitude.

Parks-McClellan optimal FIR filter design Syntax

b = firpm(n,f,a)
b = firpm(n,f,a,w)

firpm designs a linear-phase FIR filter using the Parks-McClellan algorithm. The Parks-McClellan algorithm uses the Remez exchange algorithm and Chebyshev approximation theory to design filters with an optimal fit between the desired and actual frequency responses. The filters are optimal in the sense that the maximum error between the desired frequency response and the actual frequency response is minimized. *firpm* exhibits discontinuities at the head and tail of its impulse response due to this equiripple nature.

b = firpm(n,f,a) returns row vector b containing the n+1 coefficients of the order n FIR filter whose frequency-amplitude characteristics match those given by vectors f and a. The output filter coefficients (taps) in b obey the symmetry relation:

$$b(k) = b(n+2-k), \qquad k = 1, \ldots, n+1$$

Vectors f and a specify the frequency-magnitude characteristics of the filter:

- f is a vector of pairs of normalized frequency points, specified in the range between 0 and 1, where 1 corresponds to the Nyquist frequency. The frequencies must be in increasing order.
- a is a vector containing the desired amplitudes at the points specified in f. The desired amplitude at frequencies between pairs of points (f(k), f(k+1)) for k odd is the line

segment connecting the points (f(k), a(k)) and (f(k+1), a(k+1)).The desired amplitude at frequencies between pairs of points (f(k), f(k+1)) for k even is unspecified. The areas between such points are transition or "don't care" regions.

- f and a must be the same length. The length must be an even number.

5.1.1 Filter Specifications

The implementation of Remez FIR filter is done for the specifications characterizing an equiripple FIR design method for a Band pass response type. The frequency specifications are as follows:

Sampling frequency, Fs = 48000 Hz
Stop band Frequency1. Fsotp1 = 7200 Hz
Pass band Frequency1, Fpass1 = 9600 Hz
Pass band Frequency2, Fpass2 = 12000 Hz
Stop band Frequency2. Fsotp2 = 14400 Hz

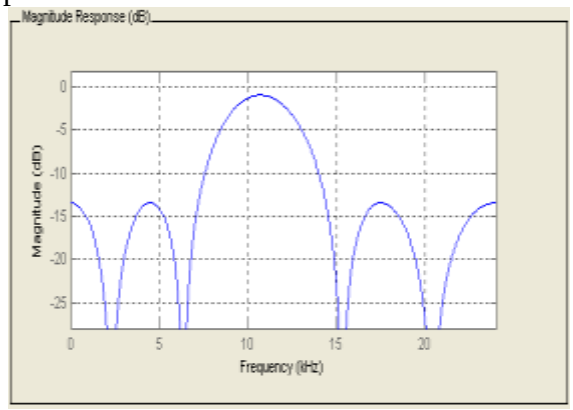

Fig 5 (a): Magnitude Response of Remez FIR filter of order 20

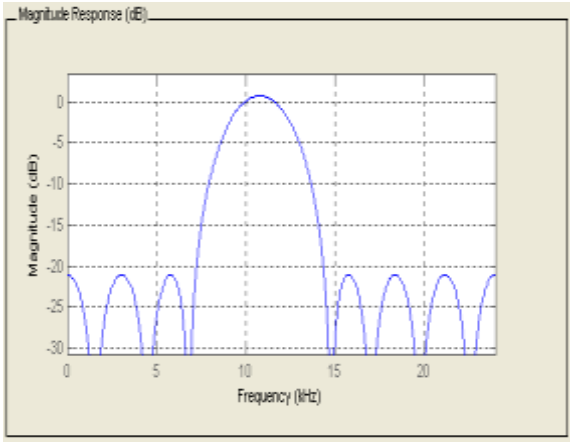

Fig 5 (b): Magnitude Response of Remez FIR filter of order 40

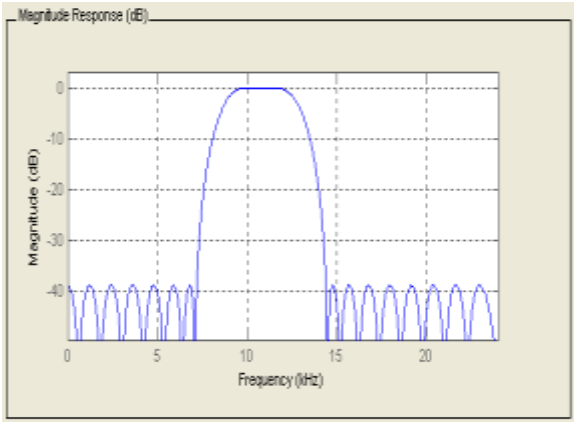

Fig 5 (c): Magnitude Response of Remez FIR filter of order 60

Using the Filter Design and Analysis Tool (fdatool) in MATLAB the filter coefficients are obtained.

5.2 VHDL Code

The following is the VHDL code for the above filter depicting the Optimal Linear Filter using the Remez Exchange Algorithm:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity sas_fir is
generic (n: integer:=4;
          m: integer:=8);
port(x:in std_logic_vector(m-1 downto 0);
          clk,rst : in std_logic;
          z : out std_logic_vector(2*m-5 downto 0));
end sas_fir;
architecture unsignd of sas_fir is
---- declaration of array for storing registers and coefficient values
type shft_reg is array (n-2 downto 0) of std_logic_vector(m-1 downto 0);
type coefficients is array (n-1 downto 0) of std_logic_vector(m-1 downto 0);
signal reg_shft: shft_reg;
signal count: std_logic_vector(21 downto 0);
signal clk1: std_logic;
signal i: std_logic_vector(7 downto 0);
signal y : std_logic_vector(2*m-1 downto 0);
------ storing coefficeint values as constant
constant w : std_logic_vector(0 to 2*m -1) := "0000000000010000";
constant coeff: coefficients := ("00010000","00100000","00110000","01000000");
begin
process(rst,clk,count)
begin
        if rst = '1' then
```

```
count <= "00000000000000000000";
                clk1 <= '0' ;
        elsif clk'event and clk = '1' then
                count <= count + 1;
if count = "11111111111111111111" then
count <= "00000000000000000000";
                clk1 <= not clk1;
        end if;
end if;
end process;
process(clk1,rst)
---------- variable declaration
variable sum : std_logic_vector(2* m - 1 downto 0):= (others=>'0');
variable o_mul: std_logic_vector(2* m - 1 downto 0):= (others=>'0');
variable sign_bit: std_logic;
------  o_mul is output of multiplication between coefficient and sample value
------  sum saves sum of two values o_mul's at different clocks
begin
------------task to be peformed at reset=1-----
if( rst='1') then
for i in n-2 downto 0 loop
for j in m-1 downto 0 loop reg_shft(i)(j)<='0';
end loop;       ------ end of loop in j
end loop;       ------ end of loop in i
------------task to be peformed at reset=0-----elsif(clk1'event and clk1='1') then
sum:= coeff(0) * x;
for i in 1 to n-1 loop
o_mul:= coeff(i) * reg_shft(n-1-i);
sum:= sum + o_mul;
end loop;       --- end of loop in i
reg_shft <= x & reg_shft(n-2 downto 1);
end if;
y <= sum; ---- sum of all the o_mul's is send as output
for i in 0 to 2*m-5 loop
z(i) <= y(i+4);
end loop;
end process;            ------- end of process
end unsignd;
```

6. Conclusions

The result of the now constructed VHDL code can has been verified on an FPGA (Field Programmable Gate Arrays).
Resources of FPGA used have been listed below:

No. of Slices : 57 out of 2352
No. of slice Flip Flops: 59 out of 4704

No. of 4-input LUTs : 96 out of 4704
No. of bonded IOBs : 22 out of 140
No. of GCLKs : 02 out of 4

A Generic FIR filter using Remez Exchange Algorithm has been implemented on a FPGA. The coefficients were obtained using the Filter Design and Analysis Tool of MATLAB. The implementation in HDL offers the possibility to generate and use the FIR filter in any developing environment. The implementation of FIR filter using Remez Exchange Algorithm finds most of its applications, particularly in industry. The FIR filter using Remez Exchange Algorithm is optimal in the sense that it minimizes the maximum error between the desired frequency response and actual frequency response. Hence, it is also called MINIMAX filter. Implementation is also found to be very much user-friendly. Besides this it gives a very universal approach to the field of signal processing. The use of Remez Exchange algorithm includes several advantages over other methods of calculating filter coefficients. This particular algorithm tremendously reduces the number of multipliers and adders being used in narrow-transition band linear-phase FIR filter. The overall process of synthesis is found to be very much faster than other methods known. It is a powerful technique for designing arbitrary magnitude linear-phase FIR filter.

References

[1] Bhasker J. (2006) Third Edition"A VHDL Primer" Pearson Education
[2] Perry L. Douglas Fourth Edition "VHDL Programming by Example" Tata Mcgraw Hill.
[3] Peter J. Ashenden, Jim Lewis (2007) " VHDL 2008-Just the New Stuff "
[4] Morgan Kaufmann, Salivahanan S., Vallavaraj A., Gnanapriya C.(2000) " Digital Signal Processing" Tata McGraw Hill
[5] Volnei A. Pedroni (2004) "Circuit Design with VHDL" MIT Press Cambridge, Massachusetts London, England.