High-Pass Digital Filter Implementation Using FPGA

Dr. Manal H. Jassim [1] **Asaad Hameed Sahar**[2]

[1] Department of Electrical Engineering, University of Technology, Baghdad
[2]Department of Computer Since, Dijla Collage, Baghdad

Email: Dr.mhalkubaisi@uotechnology.edu.iq

Received: 27/6/2013

Accepted: 12/11/2013

Abstract - **D**epending on the response of the system, digital 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 sampling 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 ripples in the pass-band and the stop-band. The frequency sampling method has the same tolerance requirements as the windowing method. In this work we implemented a digital FIR high pass filter using MATLAB program (FDATools) using sampling and windowing methods, then the design in the FPGA kit is downloaded by generating VHDL description. A comparison the amount of the component has been used in the FPGA for both methods. The FIR filter is implemented using Spartan 3AN- XC3S700a-4FG484 FPGA and simulated with the help of Xilinx ISE (Integrated Software Environment) Software WEBPACK Project Navigator 11i.

Keywords - FIR Filter, FPGA, FDATooLs

1. Introduction

The FIR filters are widely used in digital signal processing and can be implemented using programmable digital processors.

But in the realization of large order filters the speed, cost, and flexibility is affected because of complex computations.

So, the implementation of FIR filters on FPGAs is the need of the day because FPGAs can give enhanced speed, this is due to the fact that the hardware implementation of a lot of multipliers can be done on FPGA which are limited in case of programmable digital processors.

There is more than one way to implement the digital FIR filter, based on the design specification, careful choice of implementation method and tools can save a lot of time and work. Math Lab is an excellent tool to design filters.

There are toolboxes available to generate VHDL descriptions of the filters which reduce dramatically the time required to generate a solution.

Time can be spent evaluating different implementation alternatives. Proper choice of computation algorithms can improve the FPGA architecture to make it efficient in terms of speed and area [1, 2].

2. FIR Filters Design

The most interesting property of FIR filters in their linear-phase. For this property, FIR filters exhibit symmetry or anti-symmetry.

Such filters are mostly used in applications where nonlinear phase distortion cannot be tolerated.

Most standard digital signal processors have special features to efficiently implement FIR filters.

A linear-phase FIR filter is obtained by letting the impulse response exhibit symmetry around n = N/2,

i.e. $h(n) = h(N - n)$, $n = 0, 1, N$,

Or anti symmetry around n = N/2,

i.e. $h(n) = -h(N - n)$, $n = 0, 1, ..., N$,

Based on whether N is even or odd, there are four different types of FIR filters with linear phase. These types are denoted as:

Type I : $h(n) = h(N - n)$, N even
Type II : $h(n) = h(N - n)$, N odd
Type III: $h(n) = -h(N - n)$, N even
Type IV: $h(n) = -h(N - n)$, N odd

The typical impulse responses of these different types are shown in Figure.1.

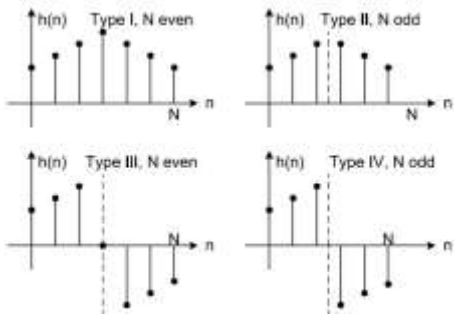

Figure .1 Linear Phase Filter Types - Impulse Response.

The center value h (N/2) is always equal to zero for Type III filters.

Furthermore, the point of symmetry is an integer corresponding to one of the samples values, in case when N is even. When N is odd, the point of symmetry lies between two sample values [3, 4].

3. FIR Coefficient Calculation Methods

The objective of most FIR coefficient calculation methods is to obtain values of h (n) such that the resulting filter meets the design specifications, such as amplitude frequency response and throughput requirements.

Several methods are available for obtaining h (n), the window and frequency sampling method are the most commonly used [5].

3.1 Window Method

In this method, use is made of the fact that $h_d(n)$ be the unit sample response of an ideal frequency selective filter with linear phase.

$$\mathbf{H_d(e^{jw}) = A(e^{jw})e^{-j(\alpha\omega-\beta)}} \quad (1)$$

Because $h_d(n)$ will generally be infinite in length, it is necessary to find an FIR approximation to $H_d(e^{jw})$.

With the window design method, the filter is designed by windowing the unit sample response,

$$\mathbf{h(n)=h_d(n)\,\omega(n)} \quad (2)$$

Where ω(n) is a finite-length window that is equal to zero outside the interval 0≤ n≤ N and is

$$\boldsymbol{\omega(n) = \omega(N - n)} \quad (3)$$

The effect of the window on the frequency response may be seen from the complex convolution theorem

$$\mathbf{H_d(e^{jw}) = \frac{1}{2\pi} H_d(e^{j\omega}) * W(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega}) W(e^{j(\omega-\theta)}) d\theta} \quad (4)$$

Thus, the ideal frequency response is smoothed by the discrete-time Fourier transform of the window, W ($\mathbf{e^{j\omega}}$).

How well the frequency response of a filter designed with the window design method approximates a desired response, $H_d(\mathbf{e^{j\omega}})$.

Is determined by two factors:

1. The width of the main lobe of W ($\mathbf{e^{j\omega}}$).
2. The peak side-lobe amplitude of W ($\mathbf{e^{j\omega}}$).

Figure (2) Main lobe, Δ, and the peak amplitude of its side lobes, A, relative to the amplitude of W ($\mathbf{e^{j\omega}}$) at ω = 0 [3].

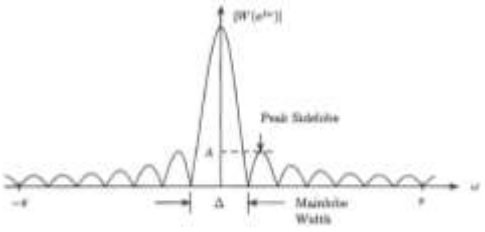

Figure.2 the DTFT of a typical window, which is characterized by the width of its

Ideally, the main-lobe width should be narrow, and the side-lobe amplitude should be small. However, for a fixed-length window, these cannot be minimized independently [1].

The filters used in this research are designed using the Math Lab FDATool using:

1- Optimal Method (also known as minimax) FIR design, which uses the Parks–McClellan and Remez Exchange methods for designing a linear-phase (symmetric) Equiripple FIR. This Equiripple design may also be used to design a differentiator or Hilbert transformer.

2- Kaiser window design using the inverse DFT method weighted by a Kaiser window.

3.2 Optimal (Equiripple) Method

A typical filter specification not only includes the specification of pass band ωp and stop band ωs frequencies and ideal gains, but also the allowed deviation (or ripple) from the desired transfer function. The transition band is most often assumed to be arbitrary in terms of ripples.

A special class of FIR filter that is particularly effective in meeting such specifications is called the Equiripple FIR.

An Equiripple design protocol minimizes the maximal deviations (ripple error) from the ideal transfer function.

The Equiripple algorithm applies to a number of FIR design instances.

The Equiripple or minimum-maximum algorithm is normally implemented using the Parks–McClellan iterative method.

The Parks–McClellan method is used to produce an Equiripple or minimax data fit in the frequency domain.

It is based on the "alternation theorem" that says there is exactly one polynomial, a Chebyshev polynomial with minimum length, that fits into a given tolerance scheme.

The length of the polynomial, and therefore the filter, can be estimated for a low pass. With

$$\mathbf{L = \frac{-10\log_{10}(\varepsilon_p \varepsilon_s) - 13}{2.324(\omega_s - \omega_p)} + 1} \quad (5)$$

Where ε_P is the pass band and ε_S the stop band ripples.

The algorithm iteratively finds the location of locally maximum errors that deviate from a nominal value, reducing the size of the maximal error per iteration, until all deviation errors have the same value.

Most often, the Remez method is used to select the new frequencies by selecting the frequency set with the largest peaks of the error curve between two iterations; this is why the Math Lab Equiripple function was called Remez in the past (now renamed to FIRPM for Parks-McClellan).

Compared with the direct frequency method, with or without data windows, the advantage of the Equiripple method is that pass band and stop band deviations can be specified differently.

This may, for instance, be useful in audio applications where the ripple in the pass band may be specified to be higher, because the ear only perceives differences larger than 3 dB [6, 7, and 8].

3.3 Kaiser Window Function

The Kaiser window is a one-parameter family of window function used for digital signal processing; Kaiser Window has very desirable characteristics both in time domain and frequency domain.

A good window should be a time limited function with a Fourier transform that is band limited and Kaiser Window possesses such characteristics closely.

Zero-th order modified Bessel function of first kind.

$$\mathbf{W[n]} = \begin{Bmatrix} \mathbf{I_0(\beta(1-\left[(n-\alpha)/\alpha\right]^2)^{1/2}\Big],\ 0 \le n \le M} \\ \mathbf{0,\ else} \end{Bmatrix} \quad (6)$$

Where $\alpha = M/2$, $I_o(.)$ is Bessel function, β is a parameter that determines to some extent the shape of the filter.

M and β trade off side lobe amplitude and main lobe width.

β controls both width and tapering off (i.e. as β increases width gets large but side lobe amplitudes get smaller) Kaiser empirically found formula going from δ, δs , δp to M and β

$$\Delta\boldsymbol{\omega} = \boldsymbol{\omega s} - \omega p \quad (7)$$

$$\mathbf{A = -20log_{10}\ \delta}$$

$$\boldsymbol{\beta} = \begin{Bmatrix} \mathbf{0.1102(A-8.7),} & \mathbf{A > 50} \\ \mathbf{0.5842(A-21)^{0.4} + 0.07886(A-21),} & \mathbf{21 \le A \le 50} \\ \mathbf{0.0,} & \mathbf{A < 2} \end{Bmatrix} \quad (8)$$

$$\mathbf{M} = \frac{\mathbf{A-8}}{\mathbf{2.285}\Delta\omega} \quad (9)$$

M does not change ripple error (to achieve δ) this is determined by the side lobe amplitude (determined by δ, or type of window).

From Figure.3 (lower) that the Equiripple design having the same tolerance requirements as the Kaiser Window design enjoys a considerably reduced filter order, i.e. 27 compared with 59[9,10].

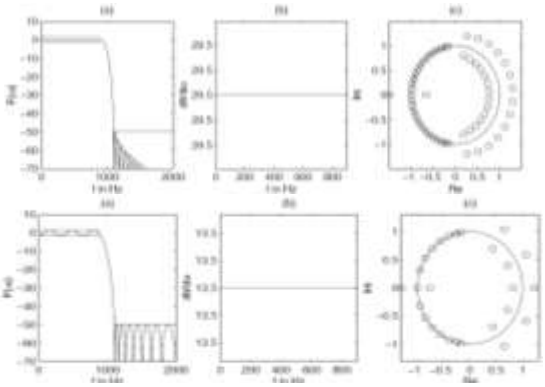

Figure.3 (upper) Kaiser Window design with L = 59, (lower) Parks-McClellan design with L = 27. (a) Transfer function. (b) Group delay of pass band. (c) Zero plots.

4. Filter Design and Analysis using FDATool

FDATool enables to design digital FIR filters by setting filter specifications, by importing filters from MATLAB workspace, or by adding, moving or deleting poles and zeros.

FDATool also provides tools for filters analysis, such as; magnitude and phase response and pole-zero plots. In this work, design of digital FIR high pass filter using FDATools using:

1) Equiripple method (The filters are set according to the following options):

Option	**Value**
Response Type	high pass
Design Method	FIR Equiripple
Filter Order	Minimum order
Options	Density Factor: 16
Frequency pacifications	Units: Hz Fs: 22050 Hz F-stop: 10000 Hz F-pass: 11000 Hz
Magnitude Specifications	Units: dB A-stop: 80 dB A-pass: 1 dB

-Filter information:
Filter structure: direct –form FIR
Filter length: 63
Stable: yes
Liner phase: yes (type 1)

-Implementation cost:
Number of multipliers = 63
Number of adders = 62
Number of states = 62
Multiplications per input sample = 63
Additions per input sample = 62

2) Kaiser Window method: (The filters are set according to the following options):

Option	**Value**
Response Type	high pass
Design Method	FIR window (Kaiser)
Filter Order	Minimum order
Frequency Specifications	Units: Hz
	Fs: 22050 Hz
	F-stop: 10000 Hz
	F-pass: 11000 Hz
Magnitude Specifications	Units: dB
	A-stop: 80 dB
	A-pass: 1 dB

-Filter information:
Filter structure: direct –form FIR
Filter length: 63
Stable: yes
Liner phase: yes (type 1)

-Implementation cost:
Number of multipliers: 63
Number of adder: 62
Number of states: 62
Multiplications per input sample: 63
Additions per input sample: 62

Table (1) show the comparing between the values of the filter impulse response coefficient h (n) for Equiripple method and Kaiser Window method

Table (1)

h(n) for Equiriple method			**h(n) for Kaiser method**	
h(0)	-0.0001942	h(61)	*0.0000131*	*h(111)*
h(1)	0.0002553	h(60)	*-0.0000234*	*h(110)*
h(2)	-0.0004150	h(59)	*0.0000370*	*h(109)*
h(3)	0.0006325	h(58)	*-0.0000541*	*h(108)*
h(4)	-0.0009191	h(57)	*0.0000744*	*h(107)*
h(5)	0.0012862	h(56)	*-0.0000970*	*h(106)*
h(10)	-0.0047004	h(51)	*-0.0001790*	*h(101)*
h(11)	0.0057543	h(50)	*-0.0001675*	*h(100)*
h(12)	-0.0069403	h(49)	*0.0001370*	*h(99)*
h(13)	0.0082558	h(48)	*-0.0000827*	*h(98)*
h(14)	-0.0096948	h(47)	*0*	*h(97)*
h(15)	0.0112478	h(46)	*0.0001155*	*h(96)*
h(20)	-0.0201184	h(41)	*-0.0012631*	*h(91)*
h(21)	0.0219475	h(40)	*0.0015985*	*h(90)*
h(22)	-0.0237278	h(39)	*-0.0019557*	*h(89)*
h(23)	0.0254283	h(38)	*0.0023230*	*h(88)*
h(24)	-0.0270179	h(37)	*-0.0026860*	*h(87)*
h(25)	0.0284665	h(36)	*0.0030270*	*h(86)*

h(30)	-0.0327167	h(30)	*-0.0036118*	*h(81)*
h(31)	0.0328461	symm etric	*0.0033246*	*h(80)*
h(50)			*0.0397538*	*h(61)*
h(51)			*-0.0420417*	*h(60)*
h(52)			*0.0439860*	*h(59)*
h(53)			*-0.0455444*	*h(58)*
h(54)			*0.0466828*	*h(57)*
h(55)			*-0.0473762*	*h(55)*
h(56)			*0.0476091*	*symmet ric*

5. Comparing the Design to Filter Specifications

FDATool allows you to measure how closely your design meets the filter specifications by using Specification masks which overlay the filter specifications on the response plot in the Display Region**.**

-- Frequency response

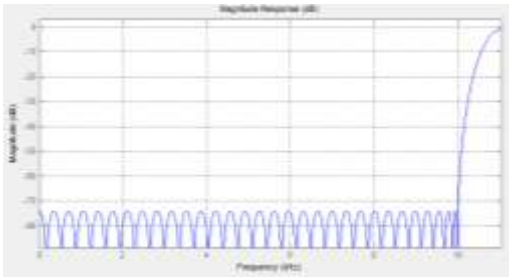

Figure .4 Frequency responses for Equiripple method.

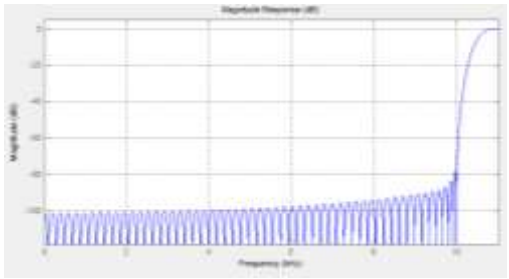

Figure.5 Frequency response for Kaiser Window method

-- Impulse response

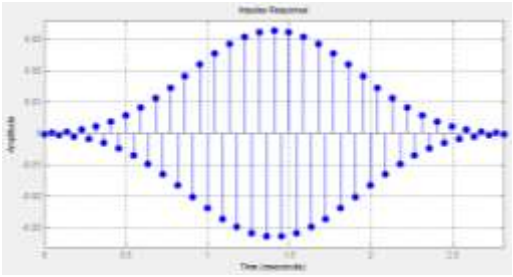

Figure.6 Impulse response for Equiripple method

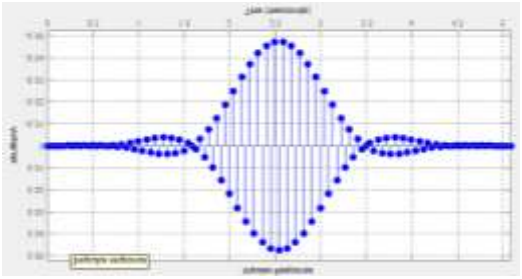

Figure.7 Impulse response for Kaiser Window method

-- Magnitude and phase responses

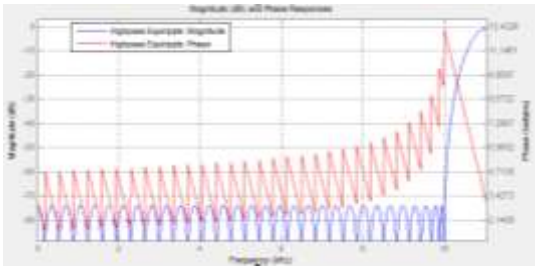

Figure.8 Magnitude and phase responses for Equiripple method

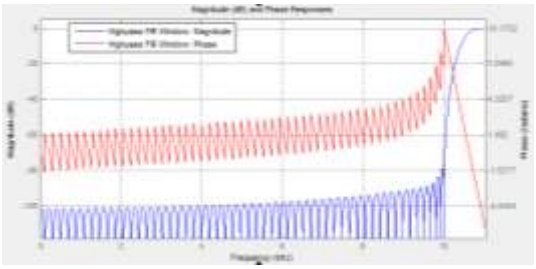

Figure.9 Magnitude and phase responses for Kaiser Window method

-- Pole and Zero plot

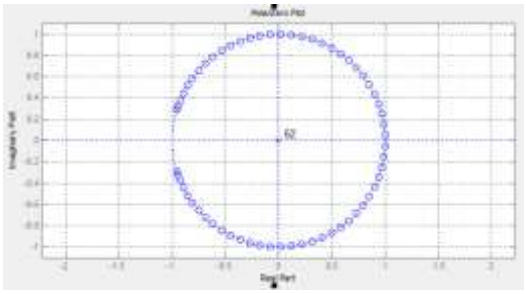

Figure.10 Pole and Zero plot for Equiripple method

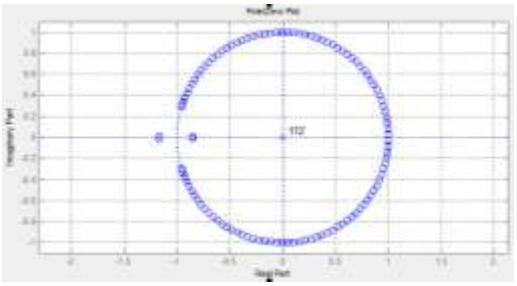

Figure.11 Pole and Zero plot for Kaiser Window method

-- Group delay

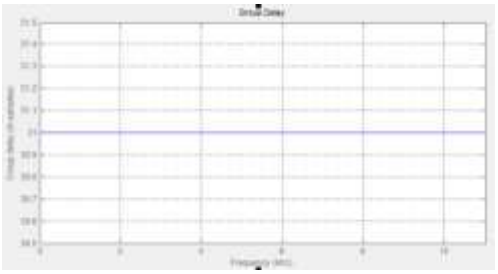

Figure.12 Group delay for Equiripple method

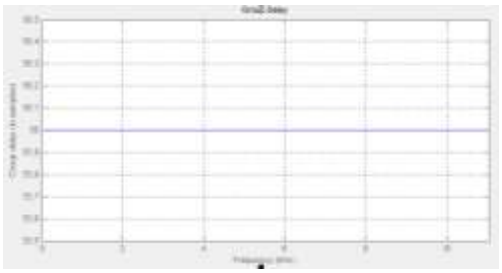

Figure.13 Group delay for Kaiser Window method

And the window viewer for Kaiser Window is shown in Figure.14.The leakage factor: (8.43%)

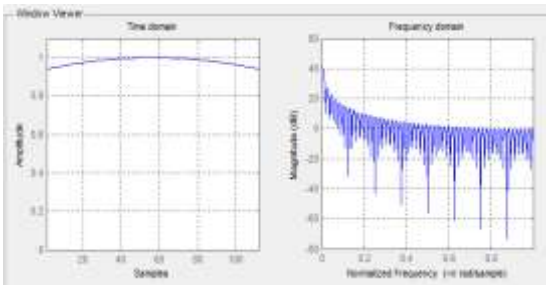

Figure.14 Window viewer

Relative side-lobe attenuation: (-13.6 dB). Main-lobe width: (-3 dB). For both design methods using type 1 linear phase because is suitable for high pass filter.

Table (2) Suitability of a given class of filter for the four FIR types

Table (2)

Type	**Low pass**	**High pass**	**Band pass**	**Band stop**
I				
II		Not suitable		Not suitable
III	Not suitable	Not suitable		Not suitable
IV	Not suitable			Not suitable

6. Implementation FIR Filter on FPGA

The program of VHDL language to download in FPGA kit has been used to enter the hardware description of FIR filter.

The implementation has been done on the FPGA platform.

The filter design has been prototyped on XC3S700-4FG484 FPGA device in Spartan-3AN Platform using ISE 11.i, after design digital filter using FDATools and translating the design to VHDL language.

From the device utilization it is found that the Kaiser window uses larger amount of resources usage in FPGA kit than Equiripple method as illustrated in Table (3) and (4).

Table (3) Device utilization summary for Equiripple method

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	816	11,776	6%	
Number of 4 input LUTs	3,210	11,776	44%	
Number of occupied Slices	3,210	5,888	54%	
Number of Slices containing only related logic	3,210	3,210	100%	
Number of Slices containing unrelated logic	0	3,210	0%	
Total Number of 4 input LUTs	3,307	11,776	48%	
Number used as logic	3,210			
Number used as a route-thru	277			
Number of bonded IOBs	35	372	9%	
Number of BUFG/BUFGCTRLs	1	24	4%	
Number of MULT18X18SIOs	20	20	100%	
Average Fanout of Non-Clock Nets	2.25			

Table (4) Device utilization summary for Kaiser Window method

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	1,808	11,776	15%	
Number of 4 input LUTs	12,300	11,776	129%	OVERMAPPED
Number of occupied Slices	8,917	5,888	151%	OVERMAPPED
Number of Slices containing only related logic	8,917	8,917	100%	
Number of Slices containing unrelated logic	0	8,917	0%	
Total Number of 4 input LUTs	16,211	11,776	137%	OVERMAPPED
Number used as logic	15,300			
Number used as a route-thru	911			
Number of bonded IOBs	35	372	9%	
Number of BUFG/BUFGCTRLs	1	24	4%	
Number of MULT18X18SIOs	20	20	100%	
Average Fanout of Non-Clock Nets	2.26			

From the tables, it can be noted that the Kaiser window uses more than actual ratio for FPGA kit (number of 4 input LUTs 129% number of occupied slices 151% total number of 4 input LUTs 137%). The RTL schematic of digital filter with main I/O is shown in Figure.15

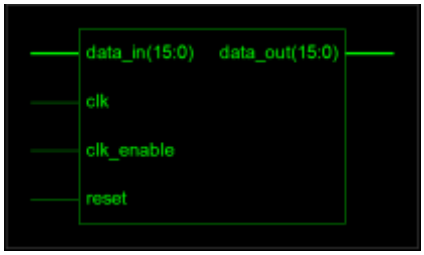

Figure.15 RTL schematic of digital filter

data_ in (15:0): is the 16-bit input vector pins
clk : is the master clock applied to the filter.
clk_enable : is the clock enable signal pin.
reset: is the reset signal pin.
data_out (15:0): is the 16-bit output vector pins.

The system must have software that can be downloaded into an FPGA.

The (VHDL) language is used to design the digital FIR filter using WebPACK™ ISE 11.i program that is downloaded the design to an FPGA development board.

In the digital FIR filter output, there are 16 output lines, so it was difficult to obtain this number on the oscilloscope, therefore, Eight Light Emitting Diodes (LED) are used to represent the real part of output.

The FDATool allows the user to generate test benches to analyses the response of the filter to a variety of test input patterns.

The HDL code for the filter designed generated using FDATools, with multipliers implemented in pipelined tree architecture.

Figure.16 show the simulation result of the FIR filter

Figure.16 Simulation result of the FIR filter

7. Conclusions

From our work we develop that Math Lab (FDATool) is an excellent tool to design filters.

There are several toolboxes available which remove the complexity for the design and implementation, toolboxes available to generate VHDL descriptions of the filters which reduce

dramatically the time required to generate a solution.

As the filter order increase the amount of resources usage increase correspondently, Non-Symmetric FIR filter took larger amount of resources than symmetric FIR filter as the amount of hardware component to implement the Non-Symmetric FIR filter is bigger than the symmetric FIR filter.

The Equiripple method is that pass-band and stop-band deviations can be specified differently, therefore, it is useful in audio applications.

The Equiripple use less amount of resources in FPGA than Kaiser Window because is decrease the order filter.

References

[1] GAO Yan and ZHANG Lin-Lin, "Simulation Study of FIR Filter Based on MATLAB", IEEE, 2010.

[2] Uwe Meyer-Baese, "Digital Signal Processing with Field Programmable Gate Arrays", Springer-Verilog Berlin Heidelberg, Third Edition, 2007.

[3] Monson H. Hayes, "Digital Signal Processing", McGraw-Hill, 1999

[4] L. Wanhammar and H. Johansson, "Digital Filters", Department of Electrical Engineering, Linkoping University, 2007.

[5] Xilinx, "Distributed Arithmetic FIR Filter", April, 2005.

[6] V. Sudhakar, N. S. Murthy and L. Anjaneyulu "Area Efficient Pipelined Architecture for Realization of FIR Filter Using Distributed Arithmetic", ICIII, 2012.

[7] Animesh Panda, Satish Kumar Baghmar, Shailesh Kumar Agrawal, T. Siva Kumar and T. Usha "FIR Filter Implementation on A FPGA Allowing Signed and Fraction Coefficients with Coefficients Obtained Using Remez Exchange Algorithm", International Journal of Advancements in Technology, 2010.

[8] H. A. Hadi, "Design and Implementation of FPGA Based Software Defined Radio Using Simulink HDL Coder", M.Sc. Thesis, College of Engineering, Department of Electrical Engineering, Al-Mustansiriya University, December, 2009.

[9] Chi-Jui Chou, Satish Mohanakrishnan and Joseph B. Evans, **"FPGA Implementation of Digital Filters"**, Telecommunications& Information Sciences Laboratory Department of Electrical & Computer Engineering, 1993

[10] R. Woods, J. McAllister, G. Lightbody and Y. Yi, "FPGA-based Implementation of Signal Processing Systems", John Wiley & Sons, Ltd., 2008.