FIR Filter Designing using Xilinx System Generator

Anurag Aggarwal Astha Satija Tushar Nagpal

Department of Electronics & Communication Engineering
Jaypee Institute of Information Technology
Sec-62,Noida, India

ABSTRACT
Xilinx System generator is used to design efficient DSP algorithm on FPGA. In this paper Finite Impulse Response (FIR) filter is designed using Simulink in Xilinx System generator. The filters have been designed using Distributed Arithmetic (DA) Algorithm. This design has been further synthesized on Xilinx Virtex-4 FPGA kit. Finally comparison is done between the results obtained from the software simulations and those from FPGA using hardware co-simulation.

Keywords
Finite Impulse Response (FIR), Distributed Algorithm (DA), Field Programmable Gate Array (FPGA), Digital Signal Processing (DSP), Analog-to-Digital Converter (ADC).

1. INTRODUCTION
Digital signal processing techniques are used extensively in a number of applications such as communication and multimedia. DSP functions such as FIR filters and transforms have numerous advantages over their analog counterparts. Digital circuits are not dependent on precise values of digital signals for their operation. Digital circuits are less sensitive to changes in component values. They are also less sensitive to variations in temperature, ageing and other external parameters. Digital processing of a signal facilitates the sharing of a single processor among a number of signals by time-sharing. This reduces the processing cost. In addition multi-rate processing is possible only in digital domain. Storage of digital data is very easy.

Digital filters are useful structures for digital signal processing applications and in signal analysis and estimation [1]. Digital filters are widely used in the world of communication and computation. An operation of digital filter design is calculation of filter transfer function coefficients that decide the response of the filter. Typical filter applications include signal preconditioning, band selection, and low/high pass filtering. Digital filters are categorized as finite impulse response (FIR) and infinite impulse response (IIR) filters. Although FIR filters are more complex, they have certain advantages over IIR filters due to which they are more widely used in filtering applications. IIR filters do not provide stability at higher orders whereas the FIR counterparts are always stable and are particularly useful for applications where exact linear phase response is required.

FIR filters [2][3] are filters having a transfer function of a polynomial in z- and is an all-zero filter in the sense that the zeroes in the z-plane determine the frequency response magnitude characteristic. The z transform of a N-point FIR filter is given by

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n} \quad (1)$$

Digital Filter can be implemented using a number of windows using the window method. Here the filter is implemented using the Blackman Window. The Blackman window is a taper formed by using the first three terms of a summation of cosines. It was designed to have close to the minimal leakage possible. The Blackman window has good (though suboptimal) characteristics for audio work. It is defined by the equation

$$\omega(n) = 0.42 - 0.5\left(\cos\frac{2\pi n}{N-1}\right) + 0.08\cos\frac{4\pi n}{N-1} \quad , \quad 0 \leq n \leq M\text{-}1 \quad (2)$$

Where M=N/2 for N even and (N+1)/2 for N odd.

Blackman window is used in signal processing literature as one of the many windowing functions for smoothing values. It is also known as an apodization (which means "removing the foot", i.e. smoothing discontinuities at the beginning and end of the sampled signal) or tapering function.

2. FIR FILTER DESIGNING
In DSP, the design methods are mainly focused in multiplier-based architectures to implement the multiply-and- accumulate (MAC) blocks that constitute the central piece in FIR filters. The FIR digital filter [2][3] is presented as:

$$y[n] = \sum_{k=0}^{N-1} c[k]x[n-k] \quad (3)$$

Where y[n] is the FIR filter output, x [n-k] is the input data and c[k] represents the filter coefficients. Equation (3) shows that multiplier-based filter implementations may become highly expensive in terms of area and speed. This issue has been partially resolved with low-cost FPGA's which use DA (Distributed Arithmetic) algorithm to implement such filters.

2.1 Distributed-Arithmetic Algorithm
DA algorithm is one of the popular multiplier-less methods which involves use of memories (RAMs, ROMs) or Look-Up Tables (LUTs) to store pre-computed values of coefficient operations. This is a powerful technique for reducing the size of a parallel hardware multiply and accumulate block that is well suited for FPGA designs. Croisier [4] had proposed the multiplier-less architecture of DA algorithm and it is based on an efficient partition of the function in partial terms using 2's complement binary representation of data. The partial terms can be pre-computed and stored in LUTs. Yoo [5] observed that the requirement of memory/LUT capacity increases exponentially with the order of the filter, given that DA implementations need 2K words, K being the number of taps of the filter.

Assuming coefficients c[k] are known constants, equation (3) can be rewritten as follows [2][3][6] :

$$y[n] = \sum_{n=0}^{N-1} c[n]x[n] \quad (4)$$

Variable x[n] can be represented in binary decimal form as follows:

$$x[n] = \sum_{b=0}^{B-1} x_b[n]2^b \qquad x_b \in \{0,1\} \quad (5)$$

where $x_b[n]$ is the b^{th} bit of x[n] and B is the input width. Finally, the inner product can be rewritten as follows:

$$y = \sum_{n=0}^{N-1} c[n] \sum_{b=0}^{B-1} x_b[k]2^b$$

$$= c[0](x_{B-1}[0]2^{B-1} + x_{B-2}[0]2^{B-2} + \cdots + x_0[0]2^0) + c[1](x_{B-1}[1]2^{B-1} + x_{B-2}[1]2^{B-2} + \cdots + x_0[1]2^0) + \cdots + c[N-1](x_{B-1}[N-1]2^{B-1} + x_{B-2}[N-1]2^{B-2} + \cdots + x_0[N-1]2^0)$$

$$= (c[0]x_{B-1}[0] + c[1]x_{B-1}[1] + \cdots + c[N-1]x_{B-1}[N-1])2^{B-1} + (c[0]x_{B-2}[0] + c[1]x_{B-2}[1] + \cdots + c[N-1]x_{B-2}[N-1])2^{B-2} + \cdots + (c[0]x_0[0] + c[1]x_0[1] + \cdots + c[N-1]x_0[N-1])2^0$$

$$y = \sum_{b=0}^{B-1} 2^b \sum_{n=0}^{N-1} c[n]x_b[k] \quad (6)$$

The coefficients in most of DSP applications for the multiply accumulate operation are constants. The partial products are obtained by multiplying the coefficients c[i] by multiplying one bit of data x[i] at a time in AND operation. These partial products are added and the result depends only on the outputs of the input shift registers. The AND functions and adders can be replaced by Look Up Tables (LUTs) [6] that gives the partial product. Input sequence is fed into the shift register at the input sample rate. The serial output is presented to the RAM based shift registers at the bit clock rate which is n+1 times (n is number of bits in a data input sample) the sample rate. The RAM based shift register stores the data in a particular address. The outputs of registered LUTs are added and loaded to the scaling accumulator from LSB to MSB and the result which is the filter output will be accumulated over the time. For an n bit input, n+1 clock cycles are needed for a symmetrical filter to generate the output. The working of DA algorithm is shown in the following figure.

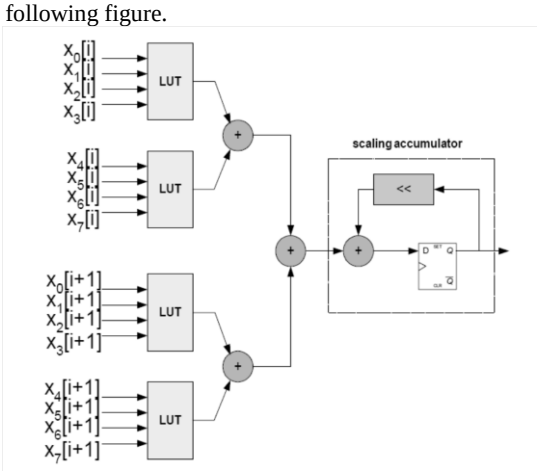


Figure 1. Block Diagram of DA Algorithm [6]

The conventional tapped delay line realization of equation (2) is shown in Figure (2). This implementation translates to L multiplications and L-1 additions per sample to compute the result. This can be implemented using a single Multiply Accumulate (MAC) [7] engine (see figure 2), but it would require L MAC cycles, before the next input sample can be processed. Thus, in a conventional MAC method with a limited number of MAC engines, as the filter length is increased, the system sample rate is decreased.

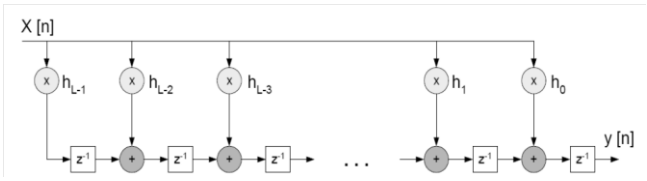


Figure 2. Block Diagram of MAC Implementation [7]

This is not the case with serial DA architectures since the filter sample rate is decoupled from the filter length. As the filter length is increased, the throughput is maintained but more logic resources are consumed.

2.2 Designing on Simulink

Simulink [8] is a block diagram environment for multi-domain simulation and Model-Based Design. It supports system-level design, simulation, automatic code generation, and continuous test and verification of embedded systems. It provides a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. It is integrated with MATLAB and enables us to incorporate MATLAB algorithms into models and export simulation results to MATLAB for further analysis. The entire design as realized in Simulink is shown in Figure (3). The constituent blocks are discussed in the proceeding text.

The *'Xilinx system generator'* [9] is a high-level tool for designing high-performance DSP systems using FPGAs. The system generator tool enables us to integrate Xilinx with Simulink, it creates a .ise file which is used in Xilinx using the model file of Simulink.

The Xilinx block sets function within gateways only i.e. *'gateway-in'* and *'gateway-out'* [8] blocks which are available in Xilinx Block set library. Any sample based input is to be passed through gateway-in block before being fed to any Xilinx block set, and then final output can be seen on *'scope'* by observing the output from gateway-out. If a frame based (music) signal is to be used as input to gateway-in, an *'unbuffer'* [8] block has to be inserted between input and gateway-in. The unbuffer block is used to convert a frame-based input to a sample-based one.

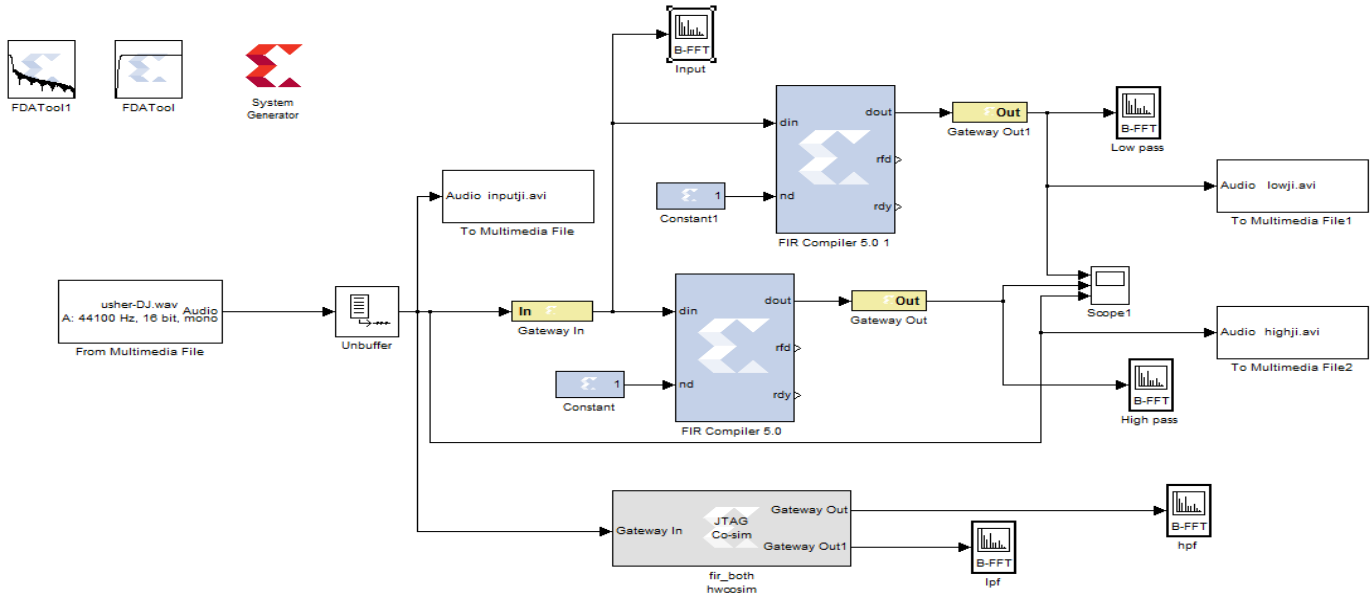
'FDA tool' [8] is the basic tool of MATLAB used to design a filter of required specifications. There are different response types (Highpass, Lowpass, Bandstop, Differentiator, Integrator etc.) and Design Methods (IIR, FIR) to implement the filter. These windows can be customized by providing order of the filter, cut-off ,sampling , pass-band and stop-band frequencies and magnitude specifications. Through the specifications provided, the tool creates coefficients that are saved as matrix in MATLAB workspace.

The major block set used in the design is *'FIR Compiler 5.0.1'* [8]. It implements a MAC and DA FIR filter as shown in Figure 4. It accepts a stream of input data and computes filtered output

with a fixed delay, based on filter configuration. The filter specification tab enables us to provide the coefficient vector as a single MATLAB row vector directly through FDA tool from the workspace.

The *'Spectrum Scope/ B-FFT'* [8] block computes and displays the mean-square spectrum or power spectral density of each signal. The signal can be a vector or a matrix. The Spectrum type parameter of the block is specified to be two-sided ((-F_s/2...F_s/2), here F_s is the sampling frequency of the original time-domain signal.

Figure 3. FIR Filter using system generator



'Hardware Co-Simulation' [8][9] block in System generator provides us with the actual routing in FPGA of the model created in Simulink. It gives the flexibility to run the design in hardware while simultaneously simulating the same in software. The generation of the Co-Simulation block guarantees that the design is synthesizable on the actual hardware used. The output from the co-simulation block can be verified with the output of the result obtained from software simulation. The table below shows the block specifications [10] for music input signal.

Table 1. FDA tool specification

High pass filter	Cut-off frequency	5Khz
	Type of Design Method	FIR-Blackman Window
	Sampling Frequency	44Khz
Low pass filter	Cut-off frequency	100Hz
	Type of Design Method	FIR-Blackman Window
	Sampling Frequency	44Khz

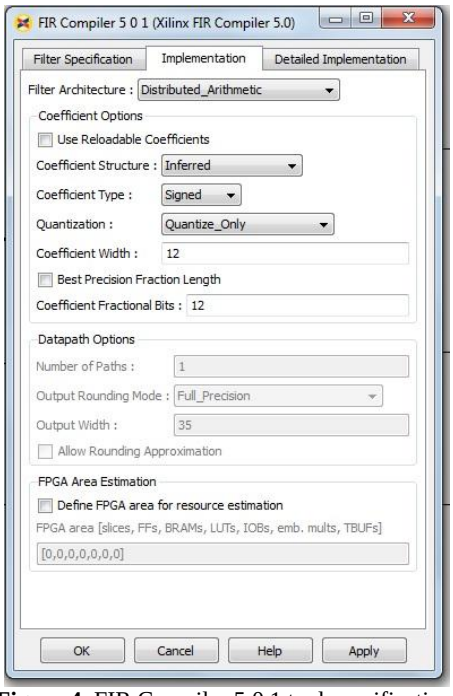


Figure 4. FIR Compiler 5.0.1 tool specification

3. FPGA

FPGAs [7][9][10] are being increasingly used for a variety of computationally intensive applications, mainly in the realm of Digital Signal Processing (DSP) and communications. Due to rapid increases in the technology, current generation of FPGAs contain a very high number of Configurable Logic Blocks

(CLBs), and are becoming more feasible for implementing a wide range of applications. The high nonrecurring engineering costs and long development time for ASICs are making FPGAs more attractive for application specific DSP solutions. The advantages of the FPGA approach to digital filter implementation include higher sampling rates than are available from traditional DSP chips, lower costs than an ASIC for moderate volume applications, and more flexibility than the alternate approaches.

The FPGA is an integrated circuit that contains many (64 to over 10,000) identical logic cells that can be viewed as standard components. Each logic cell can independently take on any one of a limited set of personalities. The individual cells are interconnected by a matrix of wires and programmable switches. A user's design is implemented by specifying the simple logic function for each cell and selectively closing the switches in the interconnect matrix. The array of logic cells and interconnects form a fabric of basic building blocks for logic circuits. Complex designs are created by combining these basic blocks to create the desired circuit. Unlike microprocessors, FPGAs are truly parallel in nature, so different processing operations do not have to compete for the same resource. Each independent processing task is assigned to a dedicated section of the chip, and can function autonomously without any influence from other logic blocks. As a result, the performance of one part of the application is not affected when more processing blocks are added.

In the process of implementing the FIR filter on FPGA, which was designed in Simulink, digital computing tasks are developed in software (Xilinx ISE 13.4) and then compiled in form of a configuration file or bit-stream that contains information on how the components should be wired together. Subsequently, a project file was imported from Simulink which was made using system generator block. This file opens as a project file in ISE software containing VHDL code for various blocks which are required to provide the necessary functionality.

Xilinx ISE project navigator provides a tool *'ChipScope Pro'* [9] which helps in observing the output of the model on the software without actually reading the output from the hardware. Signals are captured in the system at the speed of operation and brought out through the programming interface. Captured signals are then displayed and analyzed using the ChipScope Pro Analyzer tool.

As the Xilinx Virtex-4 FPGA works only on digital signals and does not contain an in-built Analog to Digital converter (ADC), an ADC is needed to be appended to the FPGA. To synchronize the clock and make FPGA and ADC work on same rising clock edge, the inbuilt 100 Mhz clock of FPGA is mitigated to 5 Mhz and fed to the ADC. A VHDL module for division of clock is added to the code. Subsequently, the code is compiled in ISE software and generate the .bit file .This bit file is then dumped into the FPGA using a platform cable which connects the software to the FPGA.

Synthesis report is used to acknowledge the count of various configuration blocks used. This is helpful in generating more efficient filters in terms of minimizing the configurable blocks, look-up tables.

4. RESULTS

The digital FIR filter was designed as described above. A music signal was given as input and the output spectrum of the high-pass and low-pass filters were observed. The results were verified using the Co-Simulation block.

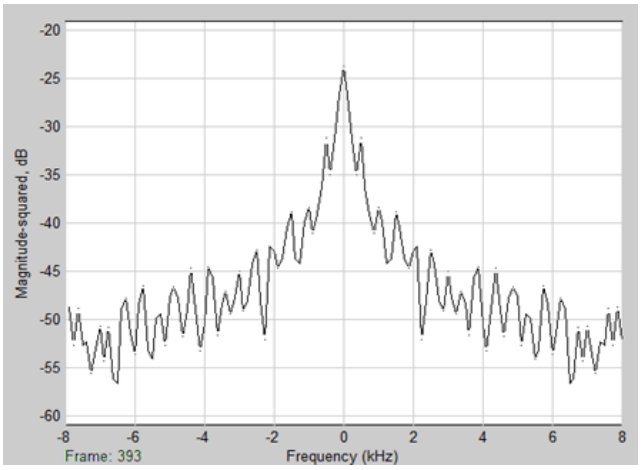


Figure 5. Input signal spectrum

The figure below show the spectrum of the input signal as well as of the outputs from highpass and lowpass filters. The Co-Simulation results have also been shown beside the results obtained from Simulink.

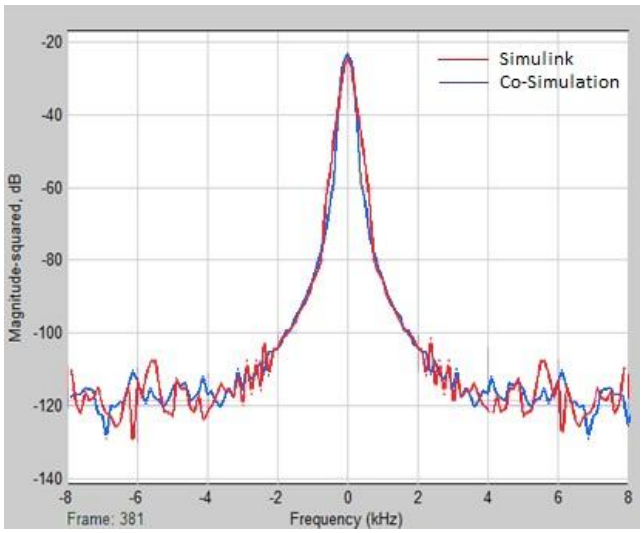


Figure 6. Output spectrum of output from Low-pass Filter

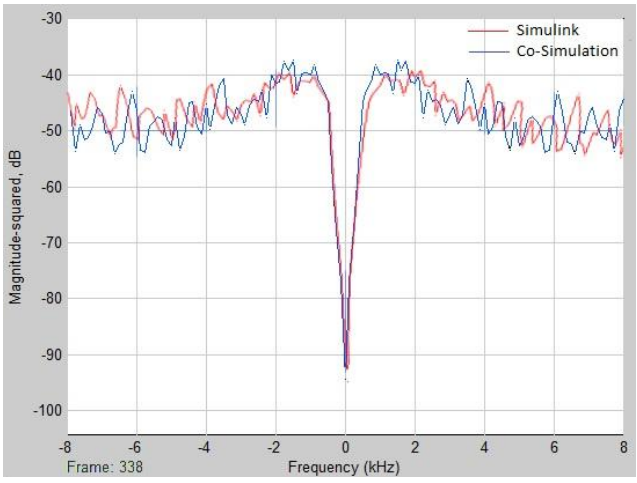


Figure 7. Output spectrum of output from High-pass Filter

The simulation result shows the close relationship of output spectrum drawn from Simulink and Hardware co-simulation. Minor differences occur due to software and practical hardware implementation.

The Xilinx ISE 13.4 also generates a synthesis report which allows us to view the results of the synthesis process. Moreover, the hardware resources which were used to implement the design on the actual FPGA kit can be observed. A portion of the report in the figure below shows the number of flip-flops, IO resources, clocks and other resources used to realize our filter design on the FPGA.

```
=========================================================================
*                      Final Report                                     *
=========================================================================
Final Results
RTL Top Level Output File Name     : fir_both_cw.ngr
Top Level Output File Name         : fir_both_cw
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : No

Design Statistics
# IOs                              : 66

Cell Usage :
# BELS                             : 2
#      GND                         : 1
#      VCC                         : 1
# FlipFlops/Latches                : 1
#      FDRE                        : 1
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 64
#      IBUF                        : 8
#      OBUF                        : 56
# Others                           : 4
#      fr_cmplr_v5_0_0555fa675eccf23b: 1
#      fr_cmplr_v5_0_8a5c910a09e011f5: 1
#      TIMESPEC                    : 1
#      xlpersistentdff             : 1
=========================================================================

Device utilization summary:
---------------------------

Selected Device : 4vsx35ff668-10

 Number of Slices:                        1  out of  15360     0%
 Number of Slice Flip Flops:              1  out of  30720     0%
 Number of IOs:                          66
 Number of bonded IOBs:                  65  out of    448    14%
 Number of GCLKs:                         1  out of     32     3%
```

Figure 8. Synthesis report of Filter design

5. CONCLUSION

This paper describes an approach towards the implementation of FIR filters using Simulink and subsequent synthesis on field programmable gate arrays (FPGA). The parallel processing capability of the FPGA greatly increases the speed of operation in the implementation of the Digital Filter. The design when simulated for music input shows best results with Blackman Window. The design provides flexibility to implement a real time digital filter which can be customized for various applications like image processing, music filtering, communications etc.

6. ACKNOWLEDGEMENTS

This research paper is made possible through the help and support of the faculty of Electronics and Communication Department at Jaypee Institute of Information Technology, Noida. We would like to extend our gratitude towards our mentor Mr. Shamim Akhter.

7. REFERENCES

[1] Jitendra Kumar Das, "Low power digital filter implementation in FPGA for hearing aid application", A thesis submitted to National Institute of Technology Rourkela

[2] Sanjit K. Mitra, " Digital Signal Processing : A Computer based approach" , McGraw Hill, 2006.

[3] A. Oppenheim and R. Schafer, "Digital Signal Processing, "Prentice-Hall, Inc., 2009.

[4] A. Croisier, D. J. Esteban, M. E. Levilion, and V. Rizo, "Digital Filter for PCM Encoded Signals", U.S. Patent No. 3,777,130, issued April, 1973.

[5] H. Yoo, and D. Anderson, "Hardware-Efficient Distributed Arithmetic Architecture for High-Order Digital Filters", in Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.

[6] Shahnam Mirzaei, Anup Hosangadi, Ryan Kastner, FPGA Implementation of High Speed FIR Filters Using Add and Shift Method, IEEE 2006.

[7] Chi-Jui Chou, Satish Mohanakrishnan, Joseph B. Evans, FPGA implementation of digital filters, Proc. of ICSPAT, 1993.

[8] MathWorks,Simulink, http://www.mathworks.com/products/simulink/

[9] Xilinx system generator, basic tutorial, www.xilinx.com

[10] Harish V. Dixit , Dr. Vikas Gupta, International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622,Vol. 2, Issue 5, September- October 2012, pp.303-307.