

Design and Implementation of Equiripple FIR Lowpass Filter on FPGA: A Case Study

Kaliprasanna Swain

*Department of Electronics and Communication Engineering
G.I.T.A, Bhubaneswar, Odisha, India*

Manoj Kumar Sahoo

*Department of Electronics and Communication Engineering
G.I.T.A, Bhubaneswar, Odisha, India*

Abstract- This paper demonstrates the design and implementation of Equiripple linear-phase FIR lowpass filter. The filter is modeled using Simulink in Xilinx System Generator. The filter Coefficients are generated with the help of FDA tools, and the System Generator tool is used for RTL code generation. Further the model is used as a filter block to interface with ADC-DAC block in VHDL. The design has been prototyped on Spartan-3 DSP prototype board XC3S500fg320 using Integrated Synthesis Environment (ISE) 13.1 tools all in one design suit from Xilinx. Finally the filters are tested by using an audio signal as input and the output is observed in CRO & speaker both.

Keywords – Equiripple Filter, FDA Tools, Simulink, Spartan-3, Xilinx System Generator, FPGA.

I. INTRODUCTION

The design of FIR filters using Windows methods leads to good performance filters. However, sometimes an FIR filter needs to be designed which not only provides good performance, but also it should be optimal. Optimization is the ability to specify a maximum error on each band of interest. This error is expressed as the absolute difference between the ideal or the desired frequency response and the actual or resulting frequency response. One of the techniques to design optimal FIR filters is to minimize a Chebyshev error criterion. The resulting filters are known as Equiripple FIR filters.

To design a linear phase Equiripple FIR filter, the Parks-McClellan and Remez exchange methods are used. For meeting the specific tolerance application Equiripple filters are mostly suited as it has equal ripple in both pass band and stop band. In general, the modified Fourier Series [1] method is used where the passband frequency response is monotonically decreasing and the maximum error is found at cutoff frequency.

But in this method, the error over the passband is spread out in an Equiripple fashion, such that at several points the maximum error is same and error can be reduced. This method is called as the *minimax* design or the *Equiripple design* where the maximum error in the passband can be minimized.

A typical filter allows the specification of passband ω_p , stopband ω_s frequencies, ideal gains, and deviation (or ripple) from the desired transfer function [2]. In terms of ripples, the transition band is most often assumed to be arbitrary. An Equiripple FIR is a special class of FIR filter which is particularly effective in meeting such specifications. The maximal deviations (ripple error) from the ideal transfer function are minimized by an Equiripple design protocol. The Equiripple algorithm applies to a number of FIR design instances. The Magnitude response of an Equiripple Lowpass filter is shown in Figure 1.

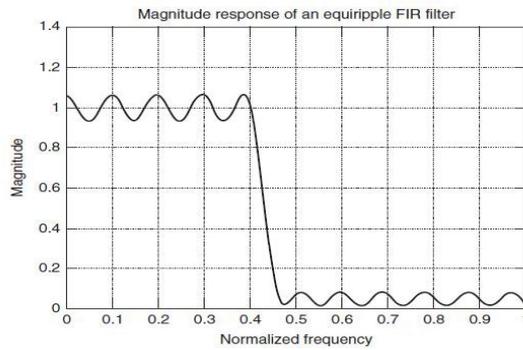


Figure 1. Magnitude response of an Equiripple lowpass filter

II. PROPOSED MODEL

The digital filter can be designed by VHDL/Verilog HDL. In this case reconfiguration method is used so that the design requires less time for any other application with minimum modification rather than fully structural recoding. This design process FIRst the fixed floating number representation is realized, then converted to an integer based representation which is more suitable for a hardware design process. The FPGA is preferred as a programmable device than any DSP or microcontroller due to memory, speed of operation and flexible architecture. The proposed model is given as in figure 2.

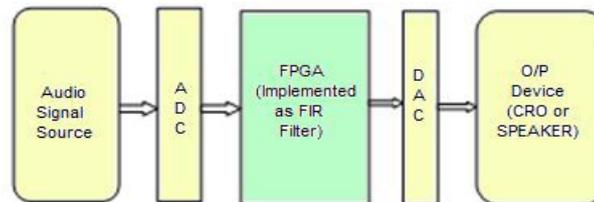


Figure 2. Purposed Model

III. FILTER IMPLEMENTATION TECHNOLOY

The details design flow of a DSP system using Xilinx System Generator FPGA is shown in figure 3. The following implementation methodology is identified. Various steps [3] [4] according to flow graph are:

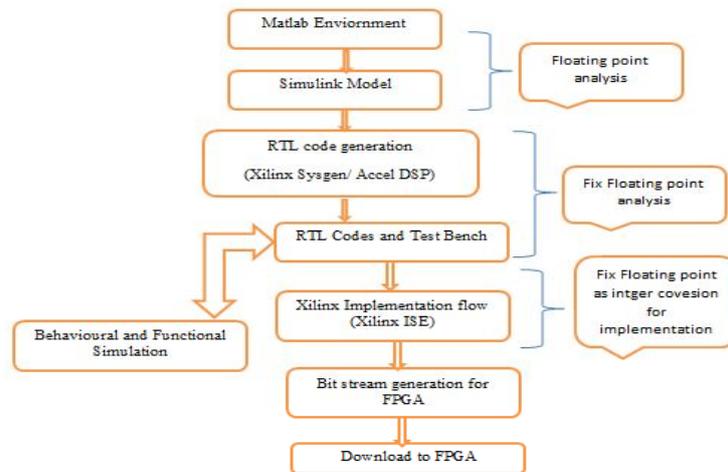


Figure 3. Design flow for DSP system design on Xilinx FPGA

Step 1: Design the Filter using FDA tool.

The Filter Design and Analysis Tool (FDA Tool) is a powerful graphical user interface (GUI) in the Signal Processing Toolbox™ for designing and analyzing digital filters of required specification. The different response types such as highpass, lowpass, bandstop, differentiator, integrator etc. and different design methods such as IIR, FIR are available to implement the filter. These windows can be customized by providing order of the filter, cutoff, sampling, passband and stop-band frequencies and magnitude specifications. By quickly setting filter performance specifications, by importing filters from MATLAB® workspace or by adding, moving or deleting poles and zeros, digital FIR and IIR filter can be designed using FDA tool. It can also be used for analyzing filters, such as magnitude and phase response plots and pole-zero plots. Design specifications for Lowpass filter using FDA tool is given in table 1.

Table -1 FDA tool specification

Response type	Lowpass
Order of filter	120
Passband Frequency, fp	5kHz
Stopband frequency, fs	10kHz
Sample frequency, fsamp	250kHz
Density factor	16

Step 2: Create Simulink Model

The Xilinx System Generator for DSP [5] [6] [7] is a system level modeling and design tool that facilitates for Xilinx FPGA design and has the ability to work at a higher level of abstraction. It enables the use of the Mathworks graphical model based Simulink design environment for FPGA design. The System Generator integrates itself with Simulink and FPGA designs are captured by using the Xilinx specific Blockset. Thus, designing a hardware model in Simulink is as simple as designing any other Simulink model with the only difference being the use of Xilinx Blockset instead of those found in Simulink. The System Generator uses the Xilinx ISE software and Xilinx IP cores generators to convert a designed model into the equivalent HDL code. The remaining FPGA implementation steps, including synthesis, place and route, etc. are automatically performed to generate a bit file that is downloaded onto the FPGA.

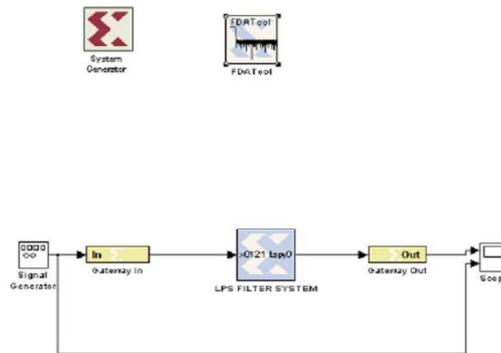


Figure 4. Top module of lowpass filter using XSG

Figure 4 shows the top module of both the filter using Xilinx System Generator.

- The gateway in blocks: The inputs into the Xilinx portion of the Simulink design considering as fixed point data in binary format, Boolean and discrete type for a digital design.
- The gateway out blocks: The outputs from the Xilinx portion of the Simulink design, it manages all digital data types with proper buffering.
- FDA tools block: The FDA tool block is used for the frequency, order parameter passing and weight generating interface with the fixed point environment which are used in later on implementation stages.
- Xilinx System generator: Xilinx System generator (XSG) is the fixed point analysis environment which can be used as integer based analysis, generation of RTL level of RTL code, co-hardware library and block for a fix FPGA IC technology.

Step 3: Implement Simulink model with the help of MATLAB.

The FIR filter is tested with a signal and the response of the filter is observed in Scope block. Here, we observe the floating point architecture, which helps to match the section given through FDA tools and later change into fixed floating point architecture by using Gateway in/out in presence of XSG. Figure 5 has shown the simulation result using Simulink.

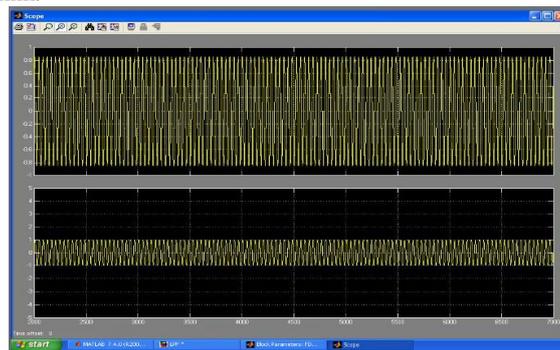


Figure 5. Netlist generation of low pass filter

Step 4: Generate the code using Xilinx System Geneator

Xilinx System Generator (XSG) as a code generator provides an equivalent synthesizable HDL code or RTL code and the test-bench for design is created with respect to the clock speed of the board and process timing, which is used as a component or sub-system in Xilinx ISE. The generation of the code depends on the block and supported IP use in the design for an FPGA family and accordingly the structure of a design is built up and interface. Fig.6 shows the code generated by XSG.

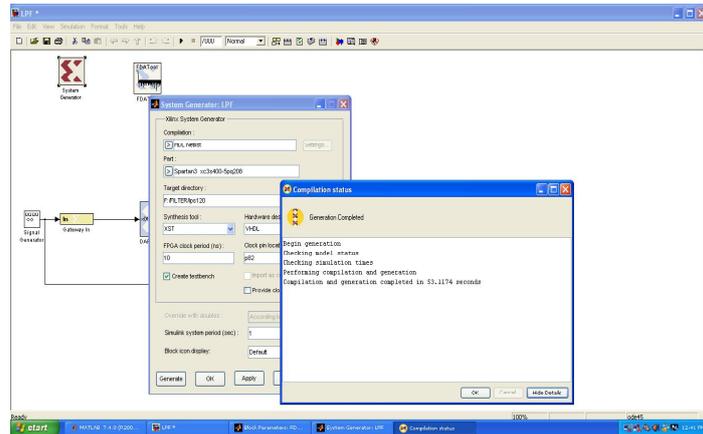


Figure 6. Netlist generation of low pass filter

Step 5: Simulate and debug the logic program and make necessary corrections to design of Step 3.

Step 6: Simulate and generate RTL schematic and check for resource utilization.

The logic program is simulated and debugged and necessary correction is made to meet the output and further RTL schematic is generated and resource utilization is checked as shown in Figure 7 and Figure 8.

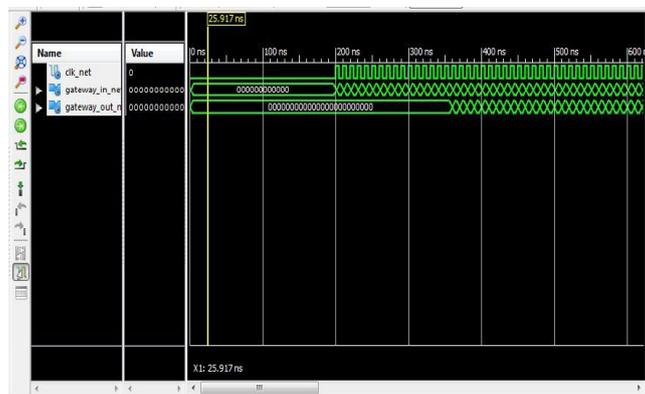


Figure 7. Lowpass filter Simulation

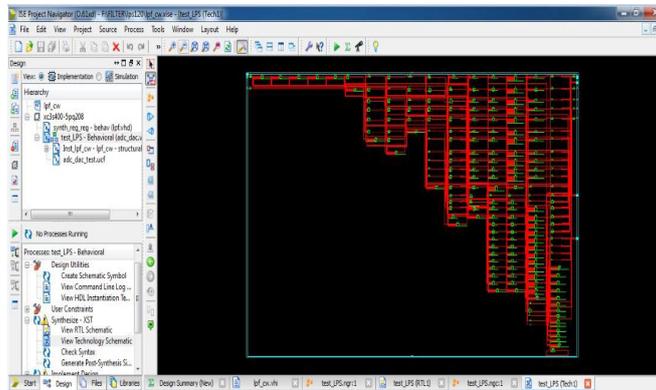


Figure 8. Lowpass RTL Design

Step 7: Run an automatic place and route program. This will place the logic block in appropriate places in FPGA and then route the interconnection between logic blocks.

An automatic place and route program was run to place the logic block in appropriate places in FPGA and then the interconnection between logic blocks were routed. Figure 9 and Figure 10 have shown the device summary and the area utility for lowpass filter.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	5,954	7,168	83%	
Number of 4-input LUTs	3,716	7,168	51%	
Number of occupied Slices	3,076	3,584	85%	
Number of Slices containing only related logic	3,076	3,076	100%	
Number of Slices containing unrelated logic	0	3,076	0%	
Total Number of 4-input LUTs	3,854	7,168	53%	
Number used as logic	3,647			
Number used as a multiplexer	136			
Number used as Shift registers	69			
Number of bonded IOBs	35	141	24%	
IOB Flip Flops	23			
Number of BUFPGMs	2	8	25%	
Average Fanout of Non-Clock Nets	1.88			

Performance Summary			
Final Timing Score:	0 (Setup: 0, Hold: 0, Component Switching Limit: 0)	Pinout Data:	Pinout Report
Routing Results:	All Signals Completely Routed	Clock Data:	Clock Report
Timing Constraints:	All Constraints Met		

Figure 9. Lowpass device summary

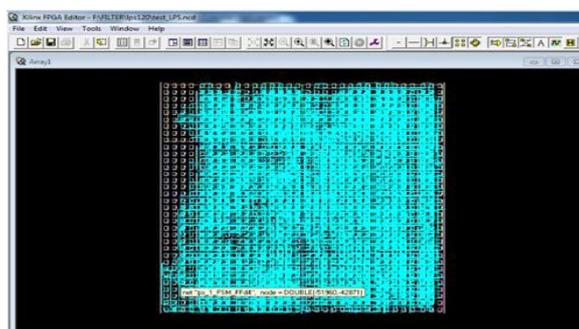


Figure 10. Area utility figure showing the route path for lowpass filter

Step 8: Generation of Bit file and then programmed it and the implementation is carried by giving an audio signal as input and the output is observed on a CRO and Speaker separately as shown in Figure 11.

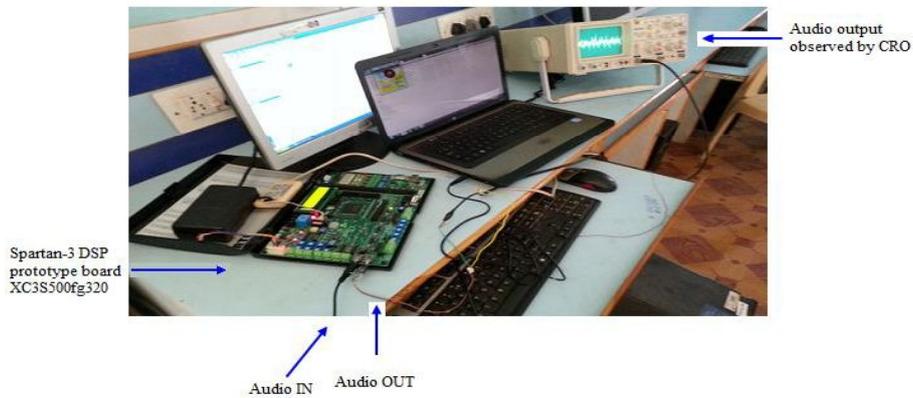


Figure 11. Experimental Setup for passing the audio signal through the filter

IV. OBSERVATION

After the implementation is carried out on Spartan-3 DSP prototype board XC3S500fg320 using Integrated Synthesis Environment (ISE) 13.1 tools, an audio signal is given as Input and the Output is observed on a CRO and speaker separately as shown in experimental setup Figure 11 and Figure 12 and 13 have shown the result when the audio signal is passed before and after the lowpass filter. By listening the output audio signal in speaker, we observed that the low tonal peak is suppressed.



Figure 12. Input test audio signal directly observed in CRO before lowpass filtering

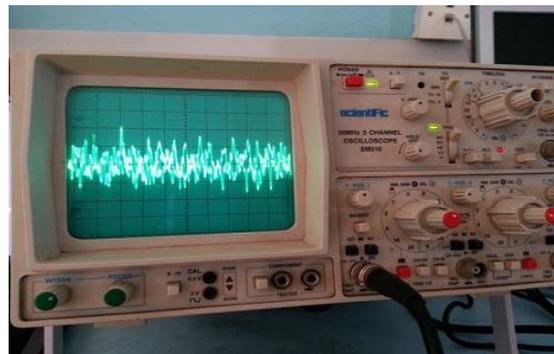


Figure 13. Output audio signal directly observed in CRO after lowpass filtering

V. CONCLUSION

This paper describes a step by step approach towards the design and implementation of Equiripple linear-phase FIR lowpass filter using Simulink in Xilinx system generator. Subsequently, it is synthesized and implemented on FPGA by using the VHDL language. It is revealed that the feasibility of designing process has greatly increased by using Matlab toolbox FDA tools when calculating the filter coefficients. Simulation results show that the FIR filter designed, fully complies with design requirements. The design is also tested by passing an audio signal and performance of the filter is checked both by loud speaker and oscilloscope. The above design can be modified or redesigned as an all pass digital filter with a single output mode or multi output mode.

REFERENCES

- [1] B. A. Sheno, Introduction to digital signal processing and filter design, JOHN WILEY & SONS, INC., pp.280- 285, 2006
- [2] UweMeyer - Baese, Digital signal processing with field programmable gate arrays, 3rd ed., Tsinghua University Press, Beijing, pp. 175 – 177, 2007
- [3] B. Mamatha1, V.V.S.V.S. Ramachandram, "Design and implementation of 120 order FIR filter based on FPGA", International Journal of Engineering Sciences & Emerging Technologies, Vol. 3, Issue. 1, 2012.
- [4] H. V. Dixit, V. Gupta, "IIR filters using Xilinx System Generator for FPGA Implementation", IJERA, Vol. 2, Issue 5, 2012.
- [5] ISE 13.2, Quick Start Tutorial, Xilinx.
- [6] Xilinx System Generator for DSP User Guide, r10.1.1, April 2008.
- [7] Nexys3™ Board Reference Manual Revision: December 28, 2011, www.digilentinc.com