 ISSN : 2230-7109 (Online) | ISSN : 2230-9543 (Print)

FPGA Implementation of IIR Filter after Checking Feasibility using Matlab & Modelsim

Manish Kansal
Dept. of ECE, Panchkula Engineering College, Barwala, Haryana, India

Abstract
Digital Signal processing ranks among the most demanding applications of digital design concepts and practices. In general, it has a rich history, and its importance is evident in diverse fields as biomedical engineering, acoustics, Sonar. The importance of the digital signal processing appears to be increasing with no visible sign of saturation. It is a mature technology domain wherein the demands for enhanced performance and reduced resource utilization have risen exponentially over the years.Digital Signal Processing (DSP) deals with the manipulation of digital signals using complex signal processing systems built from basic building blocks like filters and signal transformations.The advent of engineering tools like MATLAB has enabled the design of these basic building blocks faster and more accurate. Recent advancements in Field Programmable Gate Array (FPGA) design technology, has resulted in FPGA(s) becoming the preferred platform for evaluating and implementing signal processing algorithms. Special features of the FPGA architecture, like embedded multipliers, fast carry chains, scalability and re-configurability make it a very attractive platform for complex signal processing algorithms.This document provides a brief discourse to design, to check the feasibility of filter and implementation of digital filter algorthims based on field programmable gate arrays FPGAs).

Keywords
DSP, FIR, IIR, FPGA, Mat lab, VHDL.

I. Introduction

Digital filtering is one of the most powerful tools of DSP. A digital filter is a basic building block in any Digital Signal Processing (DSP) system. Apart from the obvious advantages of virtually eliminating errors in the filter associated with passive component fluctuations over time and temperature, op-amp drift (active filters), etc., digital filters are capable of performance specifications that would, at best, be extremely difficult, if not impossible, to achieve with an analog implementation. In addition, the characteristics of a digital filter can be easily changed under software control. Therefore, they are widely used in adaptive filtering applications in communications such as echo cancellation in modems, noise cancellation, and speech recognition

A digital filter is characterized by its transfer function. The transfer function has the form:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_M z^{-M}}$$

Where b0, b1, b2 are numerator coefficients and a0, a1, a2 are denomidator coefficients and response is calculated by dividing both coffecients.Order of the filter is N which is greater than M. There are two types of filters on basis of impulse response:

A. FIR (Finite Impulse Response): if all the coefficients a1, a2.....aM make equal to zero in equation, the filter becomes non-recursive. The difference equation for FIR filter is as follows:

$$y[n] = \sum_{k=0}^{M} b_k \, x(n-k)$$

The output of FIR filter depends only upon present and previous inputs. There is no involvement of feedback i.e. its output does not depend on previous outputs.FIR filters are output (BIBO) stable. FIR filter can be guaranteed to have linear phase. These filters also have a low sensitivity to filter coefficient quantization error. This is an important property to simple to design; they are guaranteed to be bounded input-bounded have when implementing a filter on a DSP processor or on an integrated circuit.

B. IIR (Infinite Impulse Response): The difference equation for this filter is as follows:

$$y[n] = -\sum_{k=1}^{N} a_k \, y(n-k) + \sum_{k=1}^{M} b_k \, x(n-k)$$

In this case the filter output depends upon previous inputs, present inputs and also on previous outputs. IIR filters are useful for high-speed designs because they typically require a lower number of multiply compared to FIR filters. IIR filters can be designed to have a frequency response that is a discrete version of the frequency response of an analog filter. These filters also are very sensitive to filter coefficient quantization errors that occur due to using a finite number of bits to represent the filter coefficients.

II. Designing of IIR Filter

The design of a digital filter involves following five steps:

A. Filter specification: This may include stating the type of filter, for example low pass filter, the desired amplitude and phase responses and the tolerances, the sampling frequency, the word length of the input data.

B. Filter coefficient calculation: The coefficient of a transfer function H (z) is determined in this step, which will satisfy the given specification. The choice of coefficient calculation method will be influenced by several factors. The most important of which are the critical requirements i.e. specification.

C. Realization: This involves converting the transfer function into a suitable filter network or structure

D. Analysis: The effect of quantizing the filter coefficients and input data as well the effect of carrying out the filtering operation using fixed word length on the filter performance is analyzed here.

E. Implementation: This involves producing the software code/hardware and performing the actual filtering.

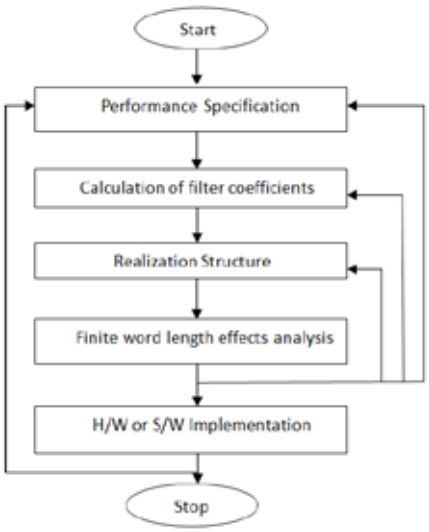

Fig.1 : Implementation of IIR filter

III. Checking IIR Filter Specifications

The specification includes
1) Signal characteristics.
2) The characteristics of filter
3) The manner of implementation.
4) Other design constraints (Cost).

The objective of the most IIR coefficient calculation methods is to obtain values of h (n) such that the resulting filter meets the design specification and to find whether it is possible to design filter with such specifications such as amplitude, frequency, response and throughput requirements. Several methods are available for obtaining h(n).The impulse invariance, optimal and bilinear transformation method are the most commonly used. We have used mat lab for checking the filter specifications. Although the above requirements are application dependent it will be helpful to devote some time on the characteristics of the filter. The characteristics of the filter are generally specified in frequency domain. For frequency selective filters, such as low pass and band pass filters. The IIR filter specifications are generally represented in terms of tolerance. The Digital Filter Design problem involves the determination of a set of filter coefficients to meet a set of design specifications. These specifications typically consist of the width of the pass band and the corresponding gain, the width of the stop band(s) and the attenuation therein; the band edge frequencies (which give an indication of the transition band) and the peak ripple tolerable in the pass band and stop band(s).
In the pass band, the magnitude response has a peak deviation of a and in the stop band, it is maximum deviation of a.The width of transition band determines how sharp the filter is. The magnitude response decreases monotonically from the pass band to stop band in this region
The following are the key parameters of interest.
$\ddot{a}^{p}$= Peak pass band deviation (or ripples)
$\ddot{a}^{s}$= stop band deviation
f_s= stop band edge frequency
f_p= pass band edge frequency
Fs=sampling frequency

The edge frequencies are often given in the normalized form, that is as the fraction of the sampling frequency (f/F)s pass band and stop band deviation may be expressed in decibels. When they specify the pass band ripples and minimum stop band attenuation respectively. Thus the minimum stop band attenuation as and the peak pass band ripple, (A) p in decibels is given as.
A_s(Stop band attenuation)=-20 $\log_{10}$ $\ddot{a}s$
A_p(Pass band ripple)= 20 $\log_{10}(1+\ddot{a}p)$

The difference between As an Ape gives the transitition width of the filter. Another important parameter is the filter length, N, which defines the number of filter. We take a set of samples a fixed time apart, and multiply them by a set of coefficients. This has an effect on the signal; by varying the coefficients we can choose what the filter does [3, 4]. The combination of the length of the filter (number of taps) and the values of the coefficients determine the filter's operation. Designing the filter is just a case of deciding how many taps and choosing the coefficients. There are many techniques for selecting coefficients. Architecture of IIR filter used has been shown below. The various blocks used in architecture of Digital IIR filter are multipliers, adders, flip flops have been shown below.

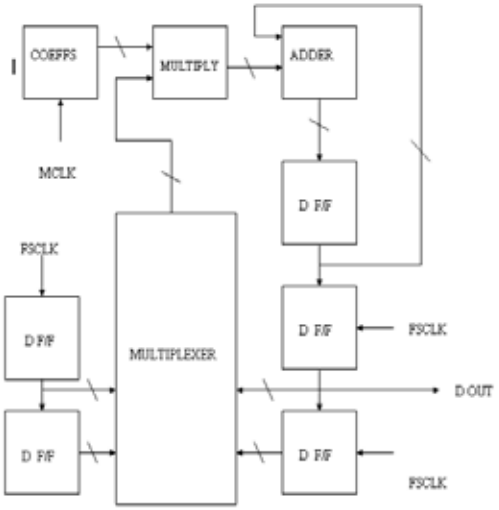

Fig. 2: Architecture of IIR filter

The schematic structure of Biquad IIR filter has been shown in fig.2 where we have used Direct Form-I structure.The way of presenting the difference equations in the form of block diagram and Signal Flow Diagram makes easy to write an algorithm, which can be implemented in the digital computer. Also the comparisons of various structures has been done which can be used for implementation of IIR filter.

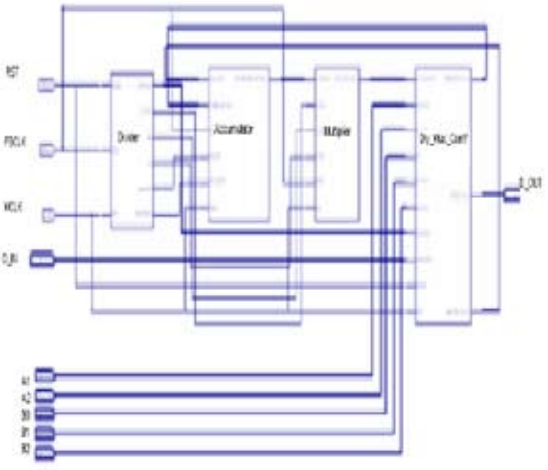

Fig.3 : Structure of IIR filter used

Table 1: Comparison of Various structures

Structure	No of Multipliers	No of additions & sub tractors	Total no of operators	wBand width
Direct	18	14	34	21
Cascade	14	16	36	20
Parallel	21	16	42	18
continue	20	19	43	28
Ladder	17	36	55	18
Wave Digital	16	32	49	20

As it is clear that Direct form requires very less no of total operators

IV. FPGA Implementation

The FPGA is advancing rapidly as a highly important element of the future of computing. Already developments have shown that it can massively reduce the price of specialized system development and it can compete on a variety of attributes with the top range commercially available microprocessors. Its initial role in rapid system prototyping is still important but in more recent times it has grown importance as a platform for implementing complete solutions. The ability to implement a fully functional system, microcontroller or even full blown computer using FPGAs and the recent advances in development software lead to highly exciting possibilities with regards to the development of complete re-configurable computing systems. There are four main categories of FPGAs currently commercially available: symmetrical array, row-based, hierarchical PLD, and sea-of-gates. In all of these FPGAs the interconnections and how they are programmed vary.

The basic FPGA architecture consists of a two-dimensional array of logic blocks and flip – flops with means for the user to configure (i) the function of each logic blocks, (ii) the inputs / outputs, and (iii) the interconnection between blocks .Families of FPGAs differ from each other by the physical means for implementing user programmability, arrangement of interconnection wires, and basic functionality of the logic blocks. Currently there are four technologies in use. They are static RAM cells, anti-fuse, EPROM transistors, and EEPROM transistors. Depending upon the application, one FPGA technology may have features desirable for that application.

A. Static RAM Technology: In the Static RAM FPGA programmable connections are made using pass transistors, transmission gates, or multiplexers that are controlled by SRAM cells.

The advantage of this technology is that it allows fast in-circuit reconfiguration. The major disadvantage is the size of the chip required by the RAM technology.

B. Anti-Fuse Technology: An anti-fuse resides in a high-impedance state and can be programmed into low impedance or “fused” state. A less expensive than the RAM technology, this device is a program- one device.

C. EPROM / EEPROM Technology: This method is the same as used in the EPROM memories. One advantage of this technology is that it can be reprogrammed without external storage of configuration; though the EPROM transistors cannot be re-programmed in-circuit

Software synthesis tools translate high-level language descriptions of the implementation into formats that may be loaded directly into the FPGAs. An increasing number of design changes through software synthesis become more cost effective than similar changes done for hardware prototypes. In addition, the implementation may be constructed on existing hardware to help further reduce the cost.

This the entire process for designing a device that guarantees that you will not overlook any steps and that you will have the best chance of getting backs a working prototype that functions correctly in your system. The design flow consists of the steps in:

Step 1: Writing a Specification

The importance of a specification cannot be overstated. This is an absolute must, especially as a guide for choosing the right technology and for making your needs known to the vendor.

As specification allows each engineer to understand the entire design and his or her piece of it. It allows the engineer to design the correct interface to the rest of the pieces of the chip. It also saves time and misunderstanding. There is no excuse for not having a specification.

A specification should include the following information:

An external block diagram showing how the chip fits into the system.

- An internal block diagram showing each major functional section.
- A description of the I/ pins including
- Output drive capability
- Input estimates including
- Timing estimates including
- Setup and hold times for input pins
- Propagation times for output pins.
- Clock cycle time
- Estimated gate count

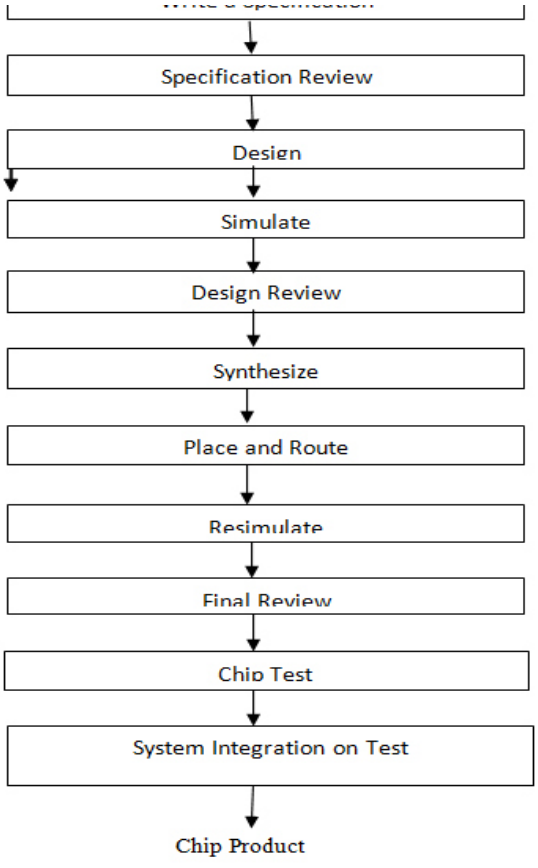

Fig. 4: FPGA Design Flow

It is also very important to understand that this is a living document. Many sections will have best guesses in them, but these will change as the chip is being designed.

Step 2: Choosing a Technology
Once a specification has been written, it can be used to find the best vendor with a technology and price structure that best meets your requirements.

Step 3: Choosing a Design Entry Method
One must decide at this point which design entry method you prefer. For smaller chips, schematic entry is often the method of choice, especially if the design engineer is already familiar with the tools. For larger designs, however, a hardware description language (HDL) such as Verilog or VHDL is used because of its portability, flexibility, and readability. When using a high level language, synthesis software will be required to "synthesize" the design. This means that the software create slow level gates from the high level description.

Step 4: Choosing a Synthesis Tool
One must decide at this point which synthesis software you will be using if you plan to design the FPGA with an HDL. This is important since each synthesis tool has recommended or mandatory methods of designing hardware so that it can correctly perform synthesis. It will be necessary to know these methods up front so that sections of the chip will not need to be redesigned later on. At the end of this phase it is very important to have a design review. All appropriate personnel should review the decisions to be certain that the specification is correct, and that the correct technology and design entry method have been chosen.

Step 5: Designing the chip
It is very important to follow good design practices. This means taking into account the following design issues.

Step 6: Simulating- Design Review
Simulation is an ongoing process while the design is being done. Small sections of the design should be simulated separately before hooking them up to larger sections. There will be much iteration of design and simulation in order to get the correct functionality. Once design and simulation are finished, another design review must take place so that the design can be checked. It is important to get others to look over the simulations and make sure that nothing was missed and that no improper assumption was made. This is one of the most important reviews because it is only with correct and complete simulation that you will know that your chip will work correctly in your system.

Step 7: Synthesis
If the design was entered using an HDL, the next step is to synthesize the chip. This involves using synthesis software to optimally translate your register transfer level (RTL) design into a gate level design that can be mapped to logic blocks in the FPGA. This may involve specifying switches and optimization criteria in the HDL code, or playing with parameters of the synthesis software in order to insure good timing and utilization.

Step 8: Place and Route
The next step is to lay out the chip, resulting in a real physical design for a real chip. This involves using the vendor's software tools to optimize the programming of the chip to implement the design. Then the design is programmed into the chip.

Step 9: Resituating – Final Review
After layout, the chip must be resituated with the new timing numbers produced by the actual layout. If everything has gone well up to this point, the new simulation results will agree with the predicted results. Otherwise, there are three possible paths to go in the design flow. If the problems encountered here are significant, sections of the FPGA may need to be redesigned. If there are simply some marginal timing paths or the design is slightly larger than the FPGA, it may be necessary to perform another synthesis with better constraints or simply another place and route with better constraints. At this point, a final review is necessary to confirm that nothing has been overlooked.

Step 10: Testing
For a programmable device, we have to simply program the device and immediately have your prototypes. You then have the responsibility to place these prototypes in your system and determine that the entire system actually works correctly. If you have followed the procedure up to this point, chances are very good that your system will perform correctly with only minor problems. These problems can often be worked around by modifying the system or changing the system software. These problems need to be tested and documented so that they can be fixed on the next revision of the chip.
However, it is possible to replace a general purpose DSP chip and design special hardware digital filters which will operate at video-speed sampling rates. In other cases, the speed limitations can be overcome by first storing the high speed ADC data in a buffer memory. The buffer memory is then read at a rate which is compatible with the speed of the DSP-based digital filter.

V. Conclusion
Low pass IIR filter gave the correct pre-synthesis and post-synthesis simulation results and requires less memory on FPGA kit as comparison to FIR .Post place and route simulation was used to find the actual delays caused by the hardware implementation of the IIR filter on FPGA. An impulse input is applied to the filter at time 45ns (this is when the system comes out of reset). The filter starts responding at time 200ns (after 55ns). The response sequence starts with the first filter coefficient and goes on till time 900ns, we found that the delay between FSCLK and MCLK is 6.5 ns and delay between input and output signal is 998396.5 ns.

VI. Results & Waveforms
While implementing a filter on hardware, biggest challenge is to achieve specified system performance at minimum hardware cost. We have designed the filter first in MATLAB in order to check the feasibility of the specifications in MATLAB. We get the desired results in MATLAB. Then the filter with the desired specifications was designed in VHDL and simulated in Modelsim software and after that burned on FPGA kit. The VHDL code of the digital IIR filter was simulated in Modelsim and the following waveforms were obtained which shows delay between FSCLK and MCLK.

ISSN : 2230-7109 (Online) | ISSN : 2230-9543 (Print)

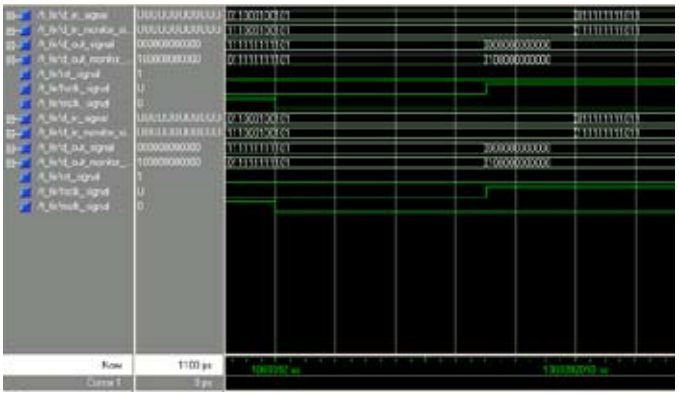

Fig. 5 : Modelsim Output

The frequency response of IIR filter has been shown with input waveforms of 100 Hz, 120Hz.

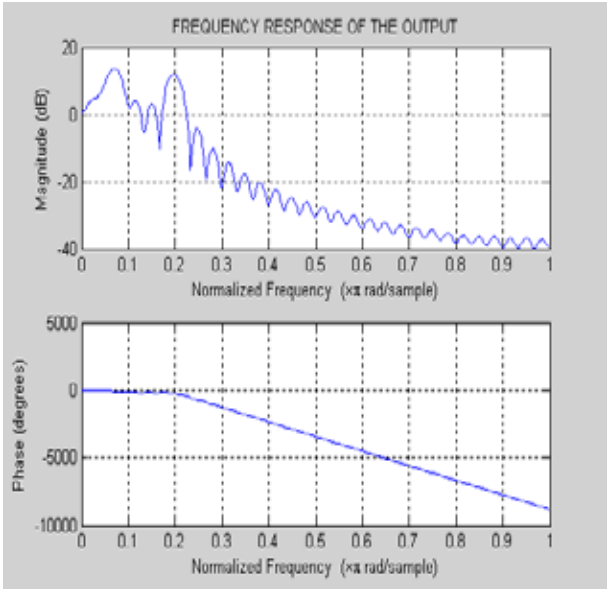

Fig. 6: Frequency response of IIR filter with 100 Hz

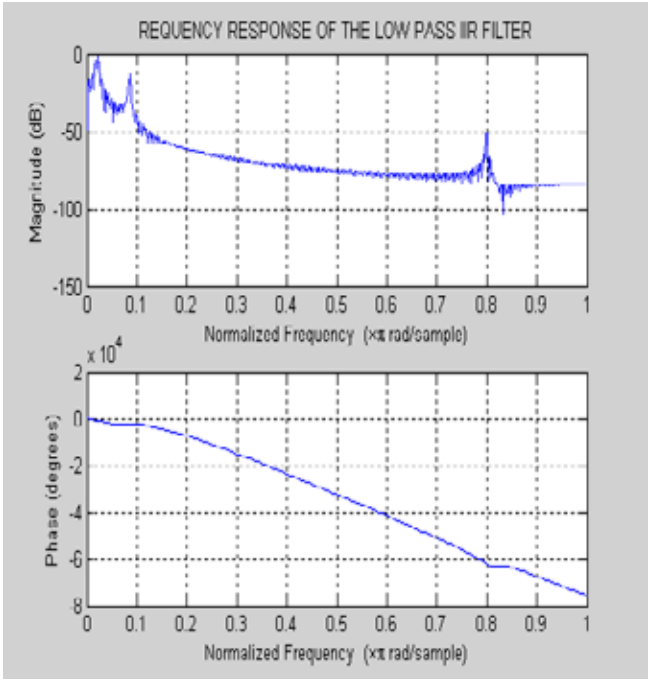

Fig.7: Frequency response of IIR filter with120 Hz

Finally FPGA kit has been burned and slots has been finalised or reserved for various components used in IIR filter. A burned FPGA kit has been shown in fig.

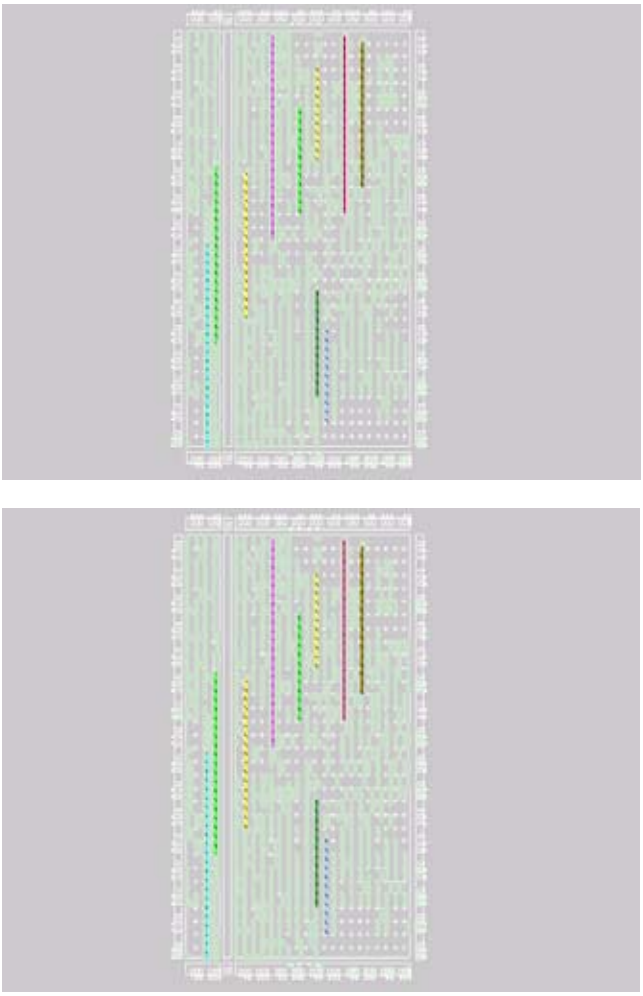

Fig. 8: FPGA kit

References

[1] ShanthalaS, S.Y.Kulkarni, "High speed and low power FPGA Implementation for DSP applications",European Journal of Scientific Research, ISSN 1450-216X, vol. 31, no.1 , pp. 19-28, 2009.

[2] Mahesh S. Chavan, R.A.Aggarwala, M.D.Uplane, "Suppression Of Baseline Wander And Power Line Interference in ECG Using Digital IIR Filter", International Journal of Circuits, Systems And Signal Processing, Issue 2,Vol. 2, 2008,pp-356-65.

[3] Appealed, B. Liu. "A New Hardware Realization of Digital Filters", IEETrans. On Acoust. Speech, Signal Process., vol. 22, pp. 456-462, 2007.

[4] Mahesh S. Chavan, R.A. Agarwala, M.D. Uplane, "Application of Chebyshev II digital filter for noise reduction in ECG Signal", WSEAStransactions on Circuits and Systems,vol. 4, no.10, pp. 1260-1267, 2005.

[5] Alarcon G, Guy CN, Binnie CD, "A simple algorithm for a electroencephalography", J Neurosci Methods. 2000.

[6] Y. C. Lim, R. Yang, D. Li, J. Song., "Signed-power-of-two term allocation scheme for the design of digital filters", IEEE Transactions on Circuits and Systems II, vol. 46, pp. 577-584, 1999.

[7] Sanjit K. Mitra James F Kaiser, "Handbook for Digital Signal Processing", John Wiley & Sons, Inc 1993.

[8] Douglas L. Perry, "VHDL", Second Edition, McGraw Hill, 1993.

Mr Manish kansal received his M.TECH in ECE from Maharishi Markandeshver University Mullana in 2011, the Btech (ECE) from Shri Krishan Institute of Engg & Technology, Kurukshetra University, in 2005. He has more than six years experience of teaching and industry. He has published many research papers in various International conferences and international Journals. His area of interests are DSP, telecommunication & Computer hardware. He is currently working as an HOD of Electronics & Communication Engineering Department in Panchkula Engineering College,Barwala.