



## **FPGA Implementation of the DIGITAL FIR Filter**

Submitted in partial fulfilment of the requirements for the award  
of degree of

### **BACHELOR OF TECHNOLOGY IN ELECTRONICS & COMMUNICATION ENGINEERING**

August 2014 - December 2014

**Submitted By:**

Ashwani Kumar Sharma  
BT11ECE005

**Submitted To:**

Mr. Sarath S.  
Assistant Professor  
Dept. of Electronics Engineering

# CERTIFICATE

This is to certify that the undergraduate project work report entitled “**FPGA Implementation of Digital FIR Filter**” submitted by **Ashwani Kumar Sharma (BT11ECE005)** as the record of the work carried out by him, is accepted as the U.G. Project Work Report submission in partial fulfilment of the requirements for the award of degree of **Bachelor of Technology in Electronics and Communication Engineering** in the **Department of Electronics Engineering**.

## **Project Guide:**

Mr. Sarath S.

Assistant Professor

Department of Electronics Engineering

Mr. Santosh Bhagat

Head (Department of Electronics)

N.I.T. Uttarakhand

# ACKNOWLEDGEMENT

The present report is the collection of the project work that has been conducted under able guidance of **Mr. Sarath S.**, Assistant Professor at **Department of Electronics Engineering** as a project guide. The work on this project has been an inspiring, often exciting, sometimes challenging, but always interesting experience. It has been made possible by many other people, who have supported me. It is a pleasure to convey my gratitude to them all in my humble acknowledgement.

In the first place I would like to record our gratitude to **Mr. Sarath S.**, for his supervision, advice and guidance from the very early stage of this project. Without his support, this project would not be completed.

I would also like to thank **Ms. Santosh Bhagat, Head Department of Electronics Engineering** who kept pushing and motivating us to perform.

Finally, I would like to thank everybody who was important to the successful realization of the project.

**Ashwani Kumar Sharma**

**BT11ECE005**

# ABSTRACT

Digital filtering algorithms are most commonly implemented using general purpose digital signal processing chips for audio applications, or special purpose digital filtering chips and application-specific integrated circuits (ASICs) for higher rates. 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. Since many current FPGA architectures are in-system programmable, the configuration of the device may be changed to implement different functionality if required.

**Keywords-** Field Programmable Gate Array (**FPGA**), Digital Signal Processors (**DSP**), Application Specific Integrated Circuit (**ASIC**), In-system Programmable (**ISP**).

# LIST OF FIGURES

| <b>Figure</b> | <b>Name</b>  | <b>Page</b> |
|---------------|--|-------------|
| 1             | DSP chip used in Guitar effects unit                         | 1           |
| 2             | Altera Cyclone II EP2C35F672C6                               | 3           |
| 3             | Schematic of clock system                                    | 4           |
| 4             | Block diagram depicting the architecture of Cyclone II board | 5           |
| 5             | Canonical Form of FIR Filter                                 | 6           |
| 6             | Interface of Simulator                                       | 7           |
| 7             | Flowchart  | 8           |
| 8             | Compilation Report   | 10          |
| 9             | RTL Netlist  | 11          |
| 10            | Vector Waveform  | 11          |

# TABLE OF CONTENTS

|                   |                                     |             |
|-------------------|-------------------------------------|-------------|
| <b>CHAPTER 1</b>  | <b>INTRODUCTION</b>                 | <b>1</b>    |
| <b>CHAPTER 2</b>  | <b>LITERATURE REVIEW</b>            | <b>2-5</b>  |
| 2.1               | Digital Filters                     | 2           |
| 2.2               | Field Programmable Gate Array       | 3-5         |
| 2.2.1             | Overview                            | 3           |
| 2.2.2             | FPGA Architecture                   | 4, 5        |
| <b>CHAPTER 3</b>  | <b>MODELLING AND IMPLEMENTATION</b> | <b>6-12</b> |
| 3.1               | Analytical Modelling                | 6           |
| 3.2               | Software Used                       | 7           |
| 3.3               | Flowchart                           | 8           |
| 3.4               | Simulation                          | 8-11        |
| 3.4.1             | Verilog HDL Code                    | 8-10        |
| 3.4.2             | Simulation Report                   | 10          |
| 3.4.3             | Synthesised Circuit                 | 11          |
| 3.4.4             | Result                              | 11          |
| 3.5               | Mathematical Verification           | 11          |
| 3.6               | Experimental Verification           | 12          |
| <b>CHAPTER 4</b>  | <b>CONCLUSIONS</b>                  | <b>13</b>   |
| 4.1               | Conclusion                          | 13          |
| 4.2               | Future Scope/Improvements           | 13          |
| <b>REFERENCES</b> |                                     | <b>14</b>   |
| <b>APPENDIX</b>   |                                     | <b>15</b>   |

# CHAPTER 1

## INTRODUCTION

For the early stages of work in the field of Digital Signal Processing people are often lured by the microprocessors due to various types of addressing modes offered by them for data input and processing. When the processing scenario started expanding it comes to process audio, images and audio-visual signals, demands for better performance at reduced costs has arisen. This leads to the development of Integrated Circuits (IC's) through Very Large Scale Integration (VLSI) with a bit different architecture customized to perform such high end processing. This leads to the development of the Digital Signal Processors (DSP's) which are customized in hardware specially designed to perform such high end Digital Signal Processing. A Digital Signal Processor can be defined as a specialized microprocessor unit with its architecture optimized to perform Digital Signal Processing. There are standard patented DSP's which are handy in most of the signal processing tasks while there might be specialized ones developed for very special purpose not required very often. **Fig 1** show a Digital Signal Processor deployed in guitar effect unit. Generally the Digital Filtering applications are implemented upon the Digital Signal Processors or Application Specific Integrated Circuits (ASIC's). 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 architecture now available support In System Programming (**ISP**) thus we can implement the adaptive filters and completely different filters in the same FPGA. The custom VLSI approach for the implementation of the DSP lacks flexibility, adoptability and they are not reconfigurable too. FPGA can eradicate these design problems through the customized approach. They resemble to the traditional gate arrays in many respects. The limitations of FPGA include the overhead imposed by the programmability. The problems may arise due to the architecture constraint such as the limitation on the number of logic functions that can be implemented in the same logic block. The FPGA avails us the facility of rapid testing and evaluation thus the design delays introduced by the fabrication delay are alleviated [1].



**Fig 1:** DSP Chip used in Guitar effects unit.[8]

# CHAPTER 2

## LITERATURE REVIEW

According to Merriam Webster dictionary the word filter stands for a device that prevents some kind of light, sound, **electronic noise**, etc., from passing through. In Electronics and Communications when it comes to signals presence of noise in the transmitted signal or introduction of noise during transmission is ubiquitous. Thus to get back the required signal, we need to perform filtering action on it. There exists several types of filters and they can be implemented at various platforms. In the following sections filters and the implementation platforms will be discussed in detail:

### 2.1 Digital Filters:

Digital filters are used in a wide variety of signal processing applications, such as audio signal processing, spectrum analysis, digital image processing, pattern recognition and audio-visual signal processing. Digital filters eliminate a number of problems associated with their classical analog counterparts and thus are preferably used in place of analog filters. Digital filters belong to the class of discrete-time LTI (linear time invariant) systems, which are characterized by the properties of causality and stability. They can be characterized in the time domain by their unit/impulse response and in the transform domain by their transfer function. Obviously, the unit-impulse response sequence of a **causal LTI system** could be of either finite or infinite duration and this property determines their classification into either **finite impulse response (FIR)** or **infinite impulse response (IIR)** system. In the practical scenario FIR filters are implemented since there does not exist any signal which lasts to infinite sample and thus have infinite samples. The general impulse response of an  $N^{\text{th}}$  order FIR Filter is given by:

$$H(z) = a_0 + a_1 z^{-1} + \dots + a_{N-1} z^{-(N-1)} \quad \text{----- (eqn.1) [7]}$$

Where  $a_0, a_1 \dots$  are filter coefficients,  $Z^{-1}$  shows a delay element represented in the Z domain. The order,  $N$  quantifies the degree of the filtering action that can be performed with the filter and it is always equal to the number of tap delay elements required to implement the FIR filter.

There are several advantages of the FIR Digital filters over IIR Filters. They are listed below:

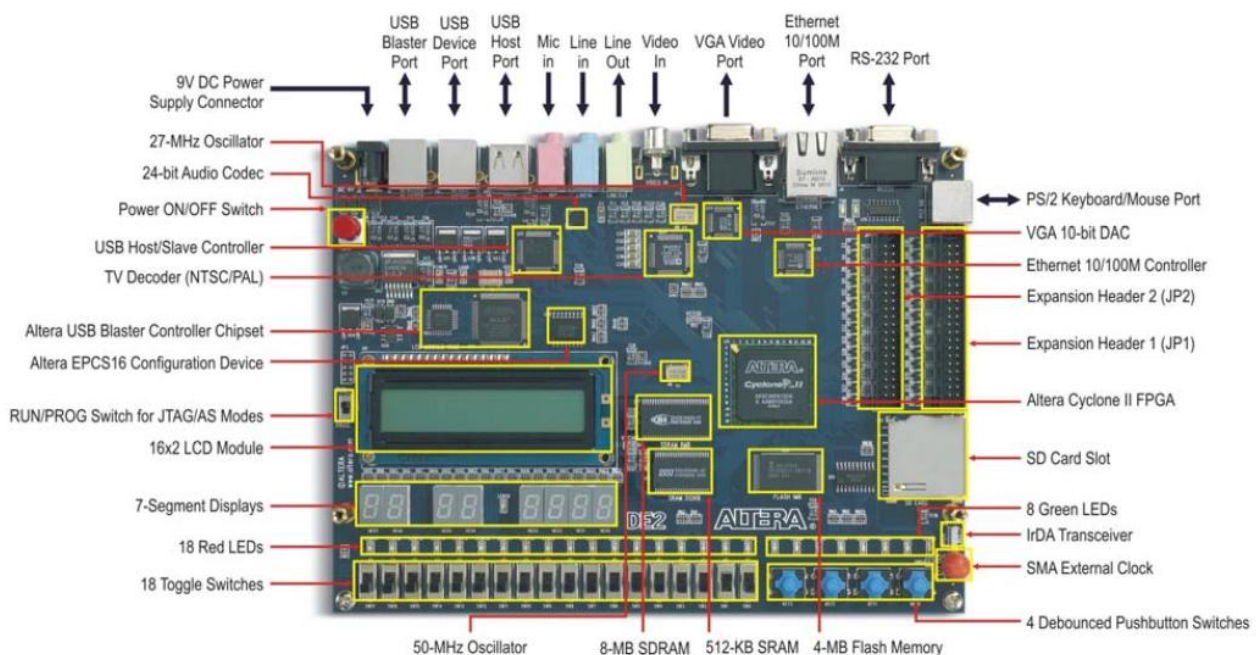
- They are always stable since their transfer function has only zeros.
- The output is only input dependent since the signal flow is unidirectional.
- The decimation and interpolation is easy and can be achieved only by varying the sampling frequency.
- They are implemented as single loop Multiply Accumulator (**MAC**) units.



## 2.2 Field Programmable Gate Array:

### 2.2.1 Overview:

The field-programmable gate array (FPGA) is a semiconductor device that can be programmed after manufacturing. Instead of being restricted to any predetermined hardware function, an FPGA allows you to program product features and functions, adapt to new standards, and reconfigure hardware for specific applications even after the product has been installed in the field—hence the name "field-programmable". You can use an FPGA to implement any logical function that an application-specific integrated circuit (ASIC) could perform, but the ability to update the functionality after shipping offers advantages for many applications. This is also known as In System Programmability means to reconfigure the device we need not to remove it from the assembly but it can be programmed there itself. Altera offers three main FPGA series: Cyclone<sup>®</sup>, Arria<sup>®</sup> and Stratix<sup>®</sup>. In the current project Cyclone II EP2C35F672C6 is being used. **Fig 2** shows the board being used.



**Fig 2:** Altera Cyclone II EP2C35F672C6. [4][5]

The specifications of the board are mentioned under:

- 33,216 Logic Elements.
- 105 M4K RAM blocks.
- 483,840 total RAM bits.
- 35 Embedded multipliers.
- 4 Phase Locked Loops.
- 475 user I/O pins.
- Fine Line Ball Grid Array 672-pin package.
- Dedicated 50 MHz and 27 MHz clocks.

The board is programmed using Altera Quartus II 9.1 Web Edition, USB Blaster. Gene rely programming is done by keeping the mode Joint Test Action Group (**JTAG**) ensuring that during trials it may not get burn on the **EEPROM** available on board although the number of erase cycles for EEPROM is very high.

There are several advantages that FPGA offers over the ASIC's and DSP's:

- Faster I/O response times and specialized functionality.
- Exceeding the computing power of digital signal processors.
- Rapid prototyping and verification without the fabrication process of custom ASIC design.
- Implementing custom functionality with the reliability of dedicated deterministic hardware.
- Field-upgradable eliminating the expense of custom ASIC re-design and maintenance.

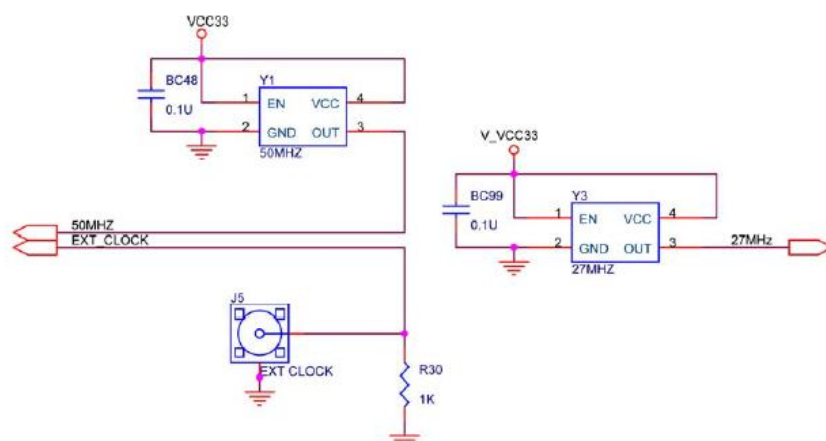
### 2.2.2 FPGA Architecture:

The architecture of FPGA varies with the vendor. Generally, every FPGA vendor provides LE, LAB, PI, Clock Circuitry, PLL, etc. The name of the units may vary with the vendors. They are discussed in brief below:

**Logic Element:** This is the smallest unit in Cyclone II FPGA architecture. It has a four-input look-up table (LUT), which is a function generator that can implement any function of four variables used for implementing the combinational logic circuits, programmable register, carry chain connector etc. The LE can operate in two modes: (i) Normal mode: This mode is used for general logic functional implementations. In this mode it supports register feedback and packed register. (ii) Arithmetic Mode: This mode is specialised for the implementation of mathematical blocks i.e. adders, multipliers, accumulators and comparators. In this mode LE can implement a 2-bit full adder and a carry chain.

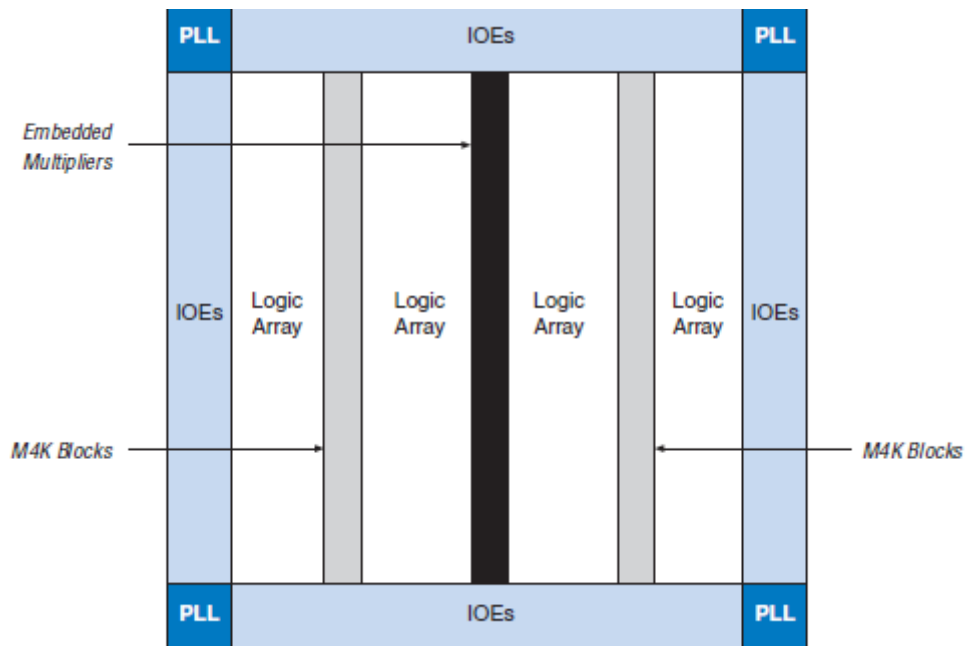
**Logic Array Block:** Every LAB consists of 16 LE's, LAB control signals, LE carry chains, local interconnects and register chains. The local interconnects are used to transfer data between LE's in same LAB. Register chain connections transfer the output of one LE's register to the adjacent LE's register within LAB.

**Clock Circuitry:** The **Cyclone II EP2C35F672C6** board has onboard **50 MHz** clock oscillator. It provides ease of 50 MHz, 27 MHz and an external clock signals which can be given as input to the ports from pins **PIN\_N2**, **PIN\_D13** and **PIN\_P16** respectively.



**Fig 3:** Schematic of clock system. [4][5]

**Fig 4** depicts the architecture of Cyclone II EP2C35F672C35:



**Fig 4:** Block diagram depicting the architecture of Cyclone II board. [4][5]

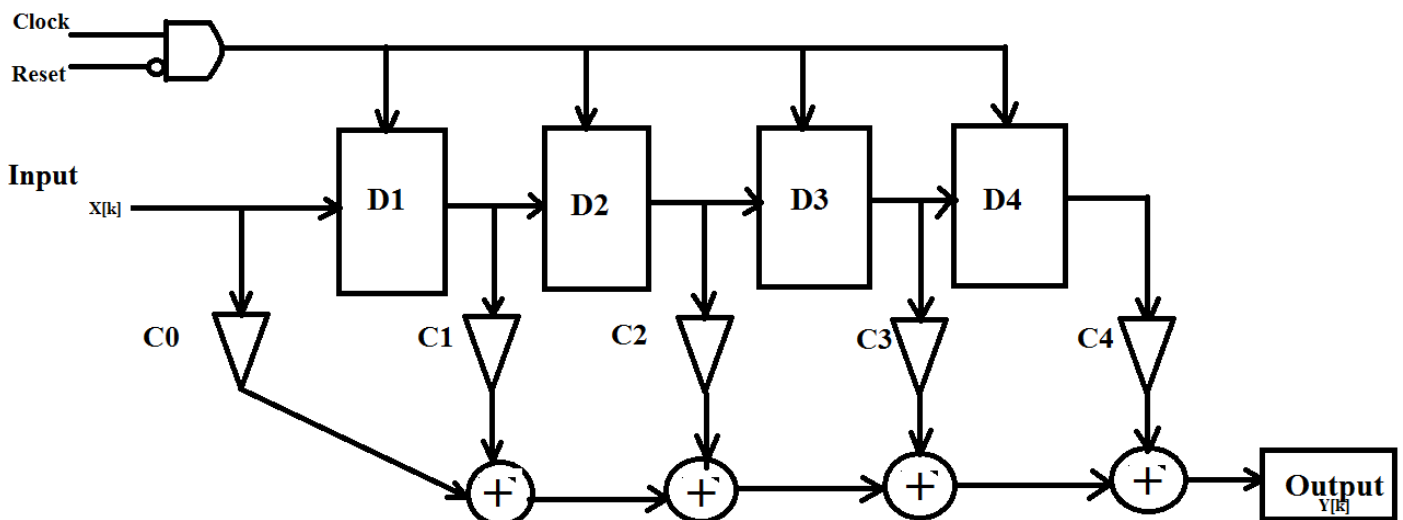
The project aims to implement a general FIR Filter with fixed point arbitrary coefficients so that one can simply put the coefficients in the program and start working with the synthesised filter. The verification will be done by carrying out mathematical calculations and then comparing the experimental results with the theoretical results [4], [5].

## CHAPTER 3

# MODELLING AND IMPLEMENTATION

### 3.1 Analytical Modelling:

Implementation of the Digital FIR Filter is done in form of the Multiplier Accumulator Units which are usually single directional data flow structures thus no feedback path is present in the FIR filters. Using Verilog Hardware Description Language (**HDL**), they can be implemented using a single loop structure. There are mainly three structures in which the FIR filters can be modelled i.e. canonical form, inverted form and pipelined form. The canonical form is most trivial out of all and the easiest of all to implement. The inverted form is just the complement of the canonical form while the pipelined form is superior of all the forms in terms of the speed but at the cost of increased complexity in implementation [2], [3]. For the sake of demonstration canonical form is implemented in this project for a fourth order filter. The five filter coefficients will be acting like gain as shown in the diagram. Fourth order filter will require four delay elements which are nothing other than the **D Flip-flops**. **Fig 5** shows the canonical form FIR filter of fourth order. The **CLOCK** and **RESET** signals are used for controlling the operation of filter. The flip flops and other components in the circuit will be triggered at **RISING** edge of the clock. The reset signal is an asynchronous signal that can be asserted by user to halt the operation of the filter. As soon as the reset is asserted the output will go to zero at the rising edge of the clock pulse. In the project the clock is supplied with **27 MHz** inbuilt clock while reset can be asserted by using a **TOGGLE SWITCH** provided on board. The input can be given either from the switches or any other external hardware and output can be obtained on the LED's.



**Fig 5:** Canonical Form of FIR Filter.

C0, C1, C2, C3, C4 and C5 are the filter coefficients.

D1, D2, D3, D4 are D flip flops, X[k] is input while Y[k] is output.

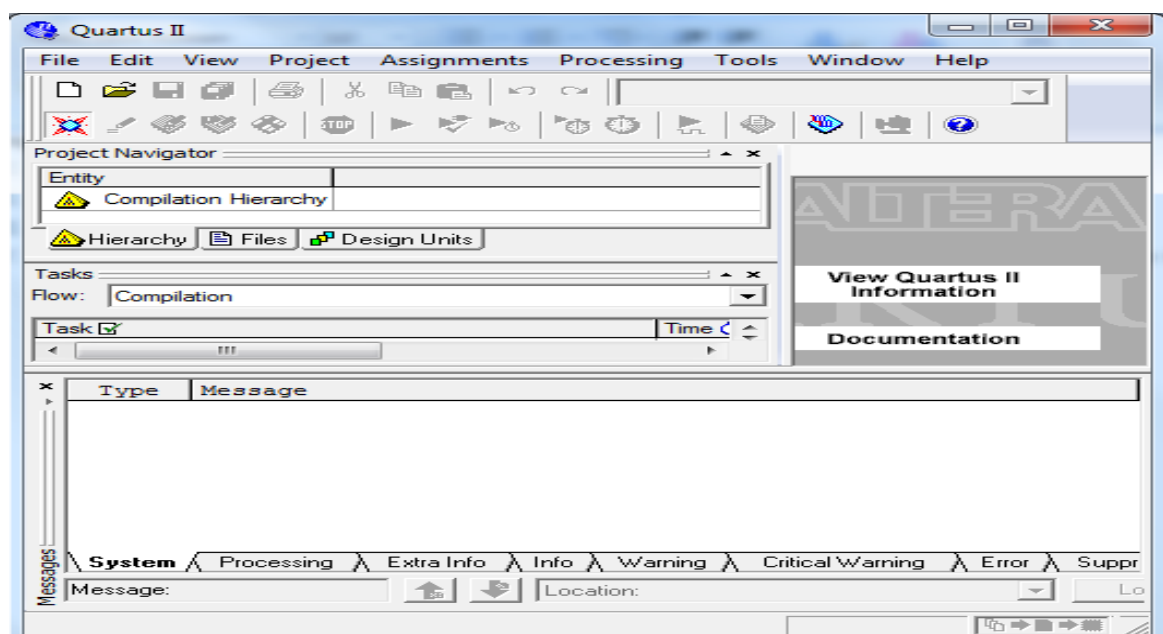
### 3.2 Software Used:

The Verilog HDL code is written in **Altera Quartus II 9.1 Web Edition**. With this we can simulate the results in terms of the waveforms. The software provides the synthesised circuit as **RTL Netlist**. The simulation can be done either making the environment Timing or Functional or a combination of both. **Fig 6** shows the Interface of the simulator. For **simulating a given circuit** following are the steps:

- I. Launch Altera Quartus II 9.1 Web Edition.
- II. Open a new Verilog HDL/ VHDL file.
- III. Type the code into the file.
- IV. Save the file but name should be same as that of the top level module. This will prompt you to create a project with the same file.
- V. Start making a project by ensuring the right working directory and make sure that device chosen must be the same you are using.
- VI. Run the program and wait for the compilation report to come.
- VII. Rectify error if any otherwise go to tools and generate the RTL Netlist.
- VIII. Open a new file and now this must be a Vector waveform one.
- IX. Insert pins and give inputs and generate functional simulation Netlist.
- X. Simulate the waveform and get the result.

#### Programming the board:

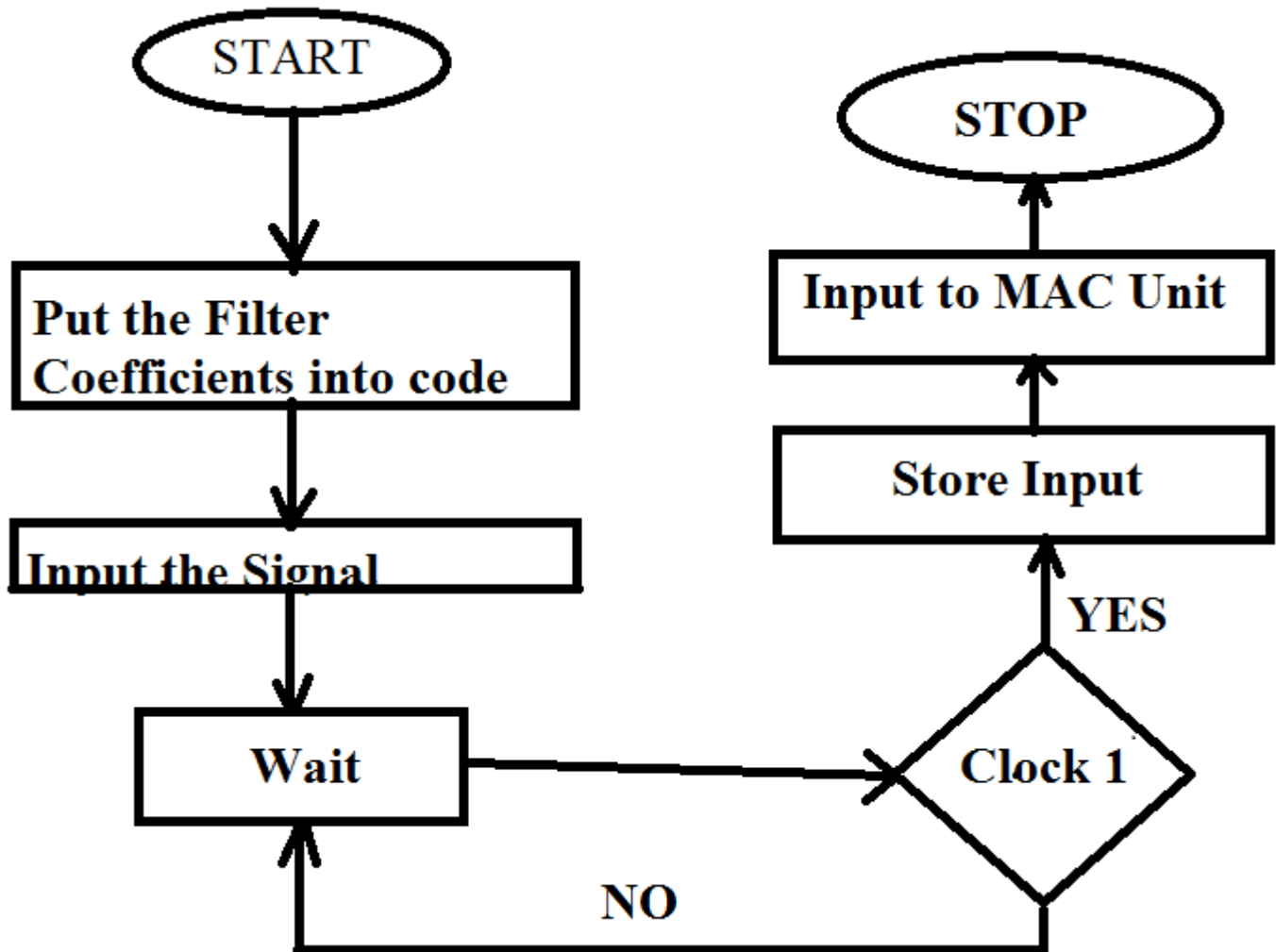
- I. Assign the pins to I/O ports.
- II. Run the program again.
- III. Click on programmer and select **.sof** file to program the board.
- IV. Before programming make sure that mode must be **JTAG** and transfer must be through **USB Blaster**.



**Fig 6:** Interface of Simulator.

### 3.3 Flowchart:

The program flow and execution can be easily understood by looking over **Fig 7** which is the flow chart of filter implementation part. The filter coefficients are supplied at the time of programming but the input data transfer to the flip-flops takes place at the rising edges of the clock pulse. The reset signal is not included for the ease of understanding.



**Fig 7:** Flowchart

### 3.4 Simulation:

The Verilog code used is discussed in this section. The comments in the code will be explaining the working of code.

#### 3.4.1 Verilog HDL Code:

/\* **fir\_filter** is the top level module which is describing the flow of data and the operations will be carried out in the program. The word size is taken as **16 bits** for the demonstration purpose. The coefficients are stored in **coeffs** identifier. The code is written using behavioural as well as data flow approach.\*/

```
module fir_filter(clk,led,reset,input_sample,output_sample);  
input reset,clk;  
input wire [15:0] input_sample;  
output reg led;  
output reg [15:0] output_sample;  
wire [3:0]hh0,hh1,hh2,hh3;  
parameter N = 5;  
reg signed[15:0] coeffs[4:0];
```

```

reg [15:0] holderBefore[4:0];
wire [15:0] toAdd[4:0];
wire clock;
reqclock (clk,clock);
D_ff d3 (clock,reset,hh2,hh3);
D_ff d2 (clock,reset,hh1,hh2);
D_ff d1 (clock,reset,hh0,hh1);
D_ff d0 (clock,reset,input_sample,hh0);

always @(*)
begin
coeffs[0]=1;
coeffs[1]=2;
coeffs[2]=2;
coeffs[3]=2;
coeffs[4]=0;
led = clock;
end
genvar i;
generate
for (i=0; i<N; i=i+1)
begin: mult
assign toAdd[i]=coeffs[i]*holderBefore[i];
end
endgenerate

always @(posedge clock or posedge reset)
begin
if(reset)
begin
holderBefore[4] <= 0;
holderBefore[3] <= 0;
holderBefore[2] <= 0;
holderBefore[1] <= 0;
holderBefore[0] <= 0;
output_sample <= 0;
end
else
begin
holderBefore[4]<=hh3;
holderBefore[3]<=hh2;
holderBefore[2]<=hh1;
holderBefore[1]<=hh0;
holderBefore[0]<=input_sample;
output_sample <= toAdd[0] + toAdd[1] +toAdd[2] + toAdd[3] + toAdd[4];
end
end
endmodule

/* D_ff is the flip flop module which is instantiated four times in the top level module fir_filter.*/
module D_ff(clk,rst,inp,out);
input clk,rst;
input wire [3:0]inp;

```

```

output reg [3:0]out;
always@(posedge clk or posedge rst)
begin
if(rst)
begin
out<=0;
end
else
begin
out<=inp;
end
end
endmodule

```

/\***reqclock** module will output the clock of required frequency while taking the on board clock of **27 MHz** as input. In this module the clock is designed to operate at **0.5 Hz** for demonstration purpose only.\*/

```

module reqclock(clk_in, clk_out);

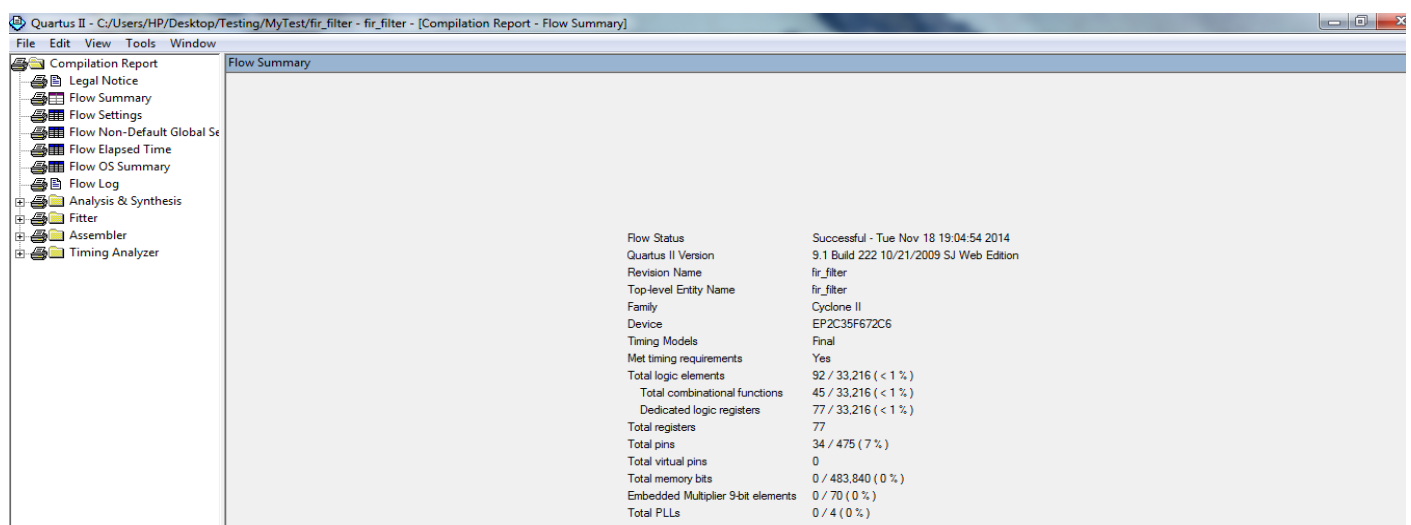
```

```

input clk_in;
output reg clk_out = 1;
parameter p = 27000000;
reg [24:0]tick = 0;
always@(posedge clk_in)
begin
if (tick < p)
begin
tick = tick+1;
end
else
begin
clk_out = ~clk_out;
tick <= 0;
end
end
endmodule

```

### 3.4.2 Simulation Report:



| Flow Summary                       |   |
|------------------------------------|---|
| Flow Status                        | Successful - Tue Nov 18 19:04:54 2014   |
| Quartus II Version                 | 9.1 Build 222 10/21/2009 SJ Web Edition |
| Revision Name                      | fir_filter                              |
| Top-level Entity Name              | fir_filter                              |
| Family                             | Cyclone II                              |
| Device                             | EP2C35F672C6                            |
| Timing Models                      | Final                                   |
| Met timing requirements            | Yes                                     |
| Total logic elements               | 92 / 33,216 ( < 1 % )                   |
| Total combinational functions      | 45 / 33,216 ( < 1 % )                   |
| Dedicated logic registers          | 77 / 33,216 ( < 1 % )                   |
| Total registers                    | 77                                      |
| Total pins                         | 34 / 475 ( 7 % )                        |
| Total virtual pins                 | 0                                       |
| Total memory bits                  | 0 / 483,840 ( 0 % )                     |
| Embedded Multiplier 9-bit elements | 0 / 70 ( 0 % )                          |
| Total PLLs                         | 0 / 4 ( 0 % )                           |

**Fig 8: Compilation Report**



### 3.4.3 Synthesised Circuit:

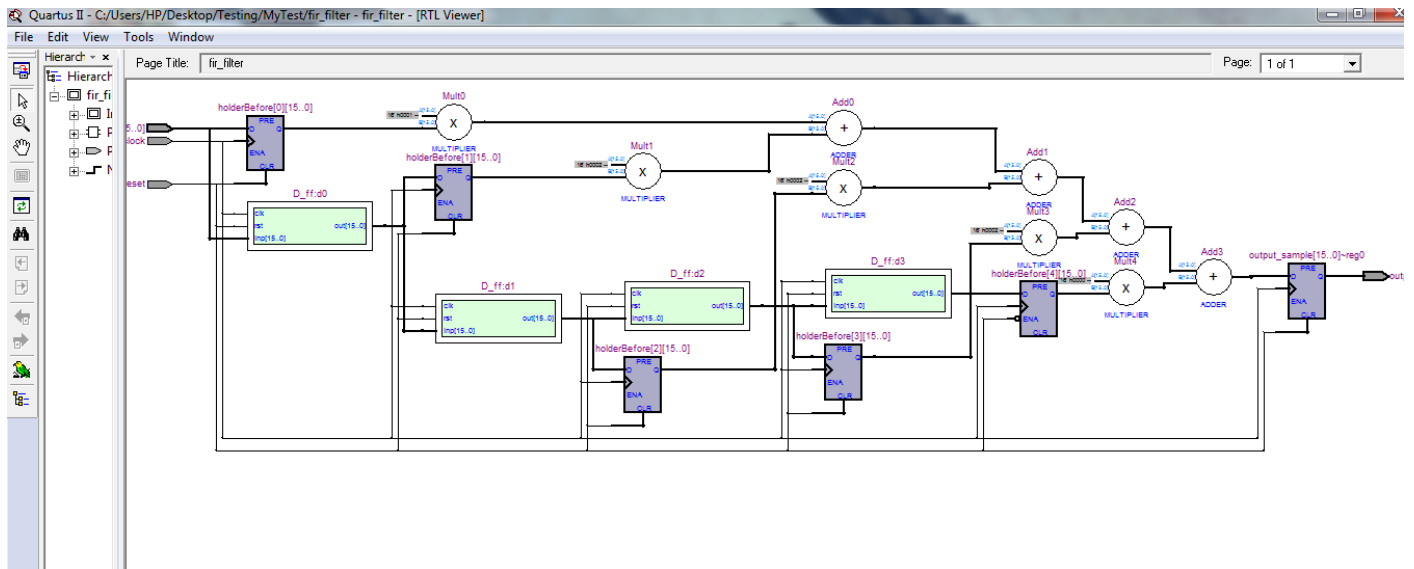


Fig 9: RTL Netlist

### 3.4.4 Result:

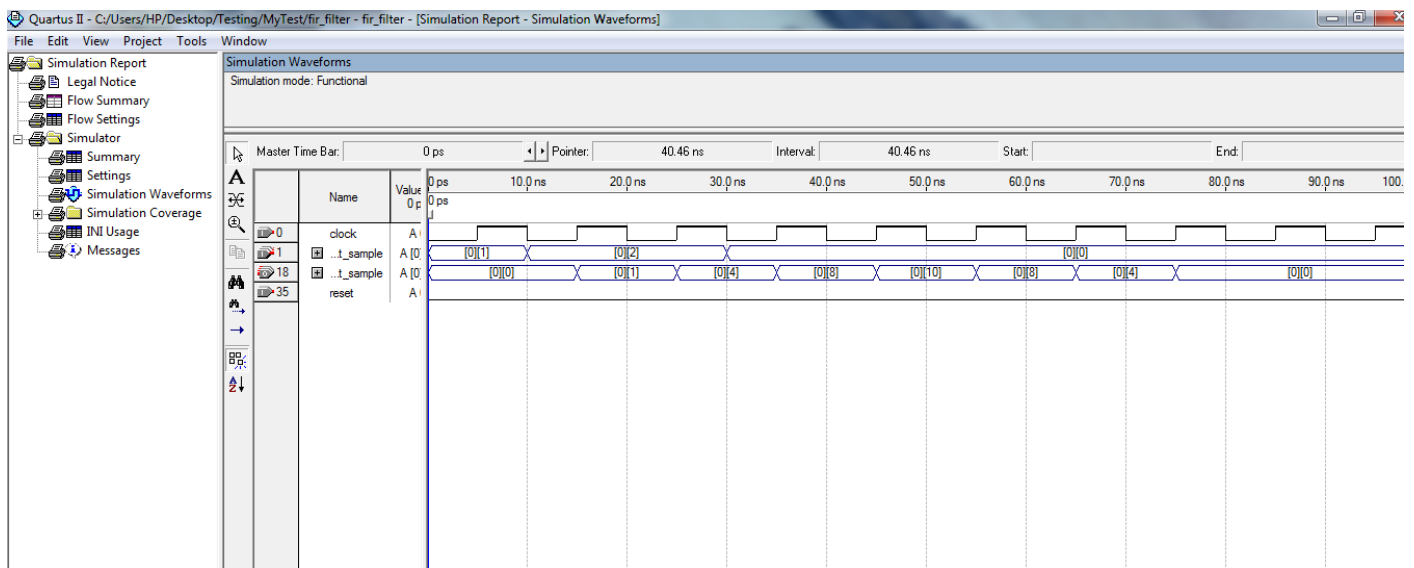


Fig 10: Vector Waveform

### 3.5 Mathematical Verification:

Let  $x[n]$ ,  $h[n]$  and  $y[n]$  represents the input signal, filter coefficients and output signals respectively. For the verification purpose the values assigned to input and coefficients are as mentioned under:

$$x[n] = \{1, 2, 2, 0\}.$$

$$h[n] = \{1, 2, 2, 2, 0\}.$$

The filter output will be the convolution of the input signal and coefficients i.e.

$$y[n] = h[n] * x[n] \quad \text{----- (eqn. 2)}$$

Where “\*” shows convolution operation. The output will be  $y[n] = \{1, 4, 8, 10, 8, 4, 0, 0\}$  which is exactly same as that obtained in the waveforms. For more details refer to Appendix.

### **3.6 Experimental Verification:**

Keeping the filter coefficients same, the same input can be given to the FPGA board using the on board and the output can be observed on the LED's present on the board. The results obtained agree with the theoretical and simulation results.

# CHAPTER 4

## CONCLUSIONS

### 4.1 Conclusion:

FPGA's are a very powerful platform for implementing Digital Signal Processing structures. They offer higher sampling rates, greater speed, and more flexibility as compared to Digital Signal Processors and Application Specific Integrates Circuits for Digital Signal Processing. The field programmability provides us ease of making changes into our implementation on the board by simply programming it within the assembly in which it is currently being used. The FPGA can be used for implementing continuously developing/altering digital signal processing structures. They act as a perfect platform for development phase of ASIC's since we can check for the performances at various stages of development on FPGA boards instead of getting it fabricated on the chips which in turn will save a lot of money that would have been wasted in the fabrication at the intermediate stages of development. Digital Filters are one of primary processing units encountered in day to day applications right from the fields of image processing, speech processing, telecommunication networks etc. A general purpose digital filter of canonical form has been implemented using Field Programmable Gate Array with random filter coefficients and result has been generated with some random input and verified mathematically. Thus, a platform for the digital signals has been created in which the operator has to input the filter coefficients and order of the filter only to get the desired filter. The synthesized filter can be used for any application by simply interfacing the external hardware with the Field Programmable Gate Array board. For interfacing the audio/ sound related hardware one need to configure the onboard available audio codec and interface the external hardware with it and similar approach can be used for the Image Processing application too.

### 4.2 Future Scope/ Improvements:

- i. The filter can be implemented with floating point numbers for the practical purposes.
- ii. For wider computational range the word size can be increased further up to floating or double precision i.e. **IEEE 754** standards **32** and **64 bit** format.
- iii. The filter can be implemented using **pipelined architecture** instead of the canonical form for higher speed in case of higher order filter for precise processing requirements.

# REFERENCES

- [1] Chi-Jui Chou, Satish Mohanakrishnan, Joseph B. Evans, 1993, "FPGA Implementation of Digital Filters", ICSPAT.
- [2] J. B. Evans, May 1993, "An efficient FIR filter architecture", IEEE international symposium Circuits and Systems.
- [3] Javier Valls, Marcos M. Piero, Trini Sansaloni, Eduardo Boemo, 1994, "A Study about FPGA based Digital Filters.
- [4] Altera, 2006, "DE2 Development and Education board user manual", Version 1.4.
- [5] Cyclone II Device Handbook Volume 1.
- [6]U. Meyer-Baese, "Digital Signal Processing with FPGA", 3rd edition Springer Berlin Heidelberg New York.
- [7]A. Oppenheim and R. Schafer, 1975, "Digital Signal Processing" .Prentice-Hall, Inc.
- [8]Wikipedia, "Digital Signal Processors", [http://en.wikipedia.org/wiki/Digital\\_signal\\_processor](http://en.wikipedia.org/wiki/Digital_signal_processor)

# APPENDIX

## 6.1 Steps used in calculating convolution:

- i. Time inverse the filter coefficients i.e.  $h[-n]$ .
- ii. Calculate  $N$  i.e.  **$N = \text{number of coefficients} + \text{number of input samples} - 1$** .
- iii. Provide a time shift of  $N$  to the time inversed  $h[n]$  i.e.  $h[N-n]$ .
- iv. Multiply this time inverted shifted version of  $h[n]$  with  $x[n]$ .
- v. Note down the individual products and sum them up.
- vi. Repeat **iii** and **iv** for all valid values of  $N$ .

## 6.2 Calculation:

For given case  $h[n]=\{1,2,2,2,0\}$  and  $x[n]=\{1,2,2,0\}$  thus  $h[-n]=\{0,2,2,2,1\}$ . The final calculated result is as under:

$$y[0]=1.1+2.0+2.0+2.0+0.0=1.$$

$$y[1]=1.2+2.1+2.0+2.0+0.0=4.$$

$$y[2]=1.2+2.2+2.1+2.0+0.0=8.$$

$$y[3]=1.0+2.2+2.2+2.1+0.0=10.$$

$$y[4]=1.0+2.0+2.2+2.2+0.1=8.$$

$$y[5]=1.0+2.0+2.0+2.2+0.2=4.$$

$$y[6]=1.0+2.0+2.0+2.0+0.0=0.$$

$$y[7]=1.0+2.0+2.0+2.0+0.0=0.$$

$$y[n]=\{1,4,8,10,8,4,0,0\}.$$