American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)
ISSN (Print) 2313-4410, ISSN (Online) 2313-4402

http://asrjetsjournal.org/

FIR Filter for Audio Signals Based on FPGA: Design and Implementation

Abdulbasit M. Sabaawi*

Communications and Signal Processing Department, Newcastle University, UK
Email: abd_albaset87@yahoo.com

Abstract

Filters play a vital role in digital signal processing (DSP) applications ranging from Video and image processing to wireless communication. In this paper, Low pass digital finite impulse response (FIR) filter is designed and implemented. Eight coefficients and taps are used in the design. The whole system are coded in VHDL language using modular design approach and implemented in Altera DE1 board. This board has cyclone II Field Programmable Gate Array (FPGA), Codec chip (WM8731), Embedded DSP multipliers and embedded processor support. The design is implemented as three main blocks: Codec initialization block, serial to parallel (S2P) Adapter block and FIR filter block. The blocks are tested and simulated in order to ensure that the result is correct. Finally, the Quartus II software tool is used to evaluate the implementation results and obtain the frequency response of the designed filter.

Keywords: FIR Filters; Digital Filters; FPGA; I2C Protocol; Codec chip (WM8731).

1. Introduction

In signal processing, the filter is a device that rejects unwanted parts of the signal, such as random noise, or extracts useful range of frequencies [1]. The function of an ideal filter is to allow only certain frequencies to pass whereas blocking all others. This depends on the cut off frequency of required filter. Filters are characterized as: Low pass, high pass, band pass and band stop filters. One example of electric filters is those that being used in television and radio set by passing its band of frequencies into a certain channel while removing out those of other channels [2]. FIR filter also known as non-recursive filter is commonly used in signal processing.

---

* Corresponding author.

The output of FIR filter depends on the present and previous inputs. This will make it stable and easy to implement. In other words, FIR filters are more realizable in hardware implementation [3]. Filter with a large number of coefficients and taps is essential to increase the performance depending on the hardware requirements [4].

The equations below show the frequency response of FIR filter in the z- domain and the relationship between y(n) and x(n).

$$y(n) = \sum_{k=0}^{N-1} b_k\, x(n-k)\ , \qquad H(z) = \sum_{k=0}^{N-1} b_k\, Z^{-k} \tag{1}$$

It can be observed from the equations that $\mathbf{b_k}$ presents the coefficients of the filter. Moreover, there is no feedback in FIR filter and the internal structure of this filter can be represented by series of block diagrams, called tapped delay line or direct realisation as demonstrated in Figure 1.

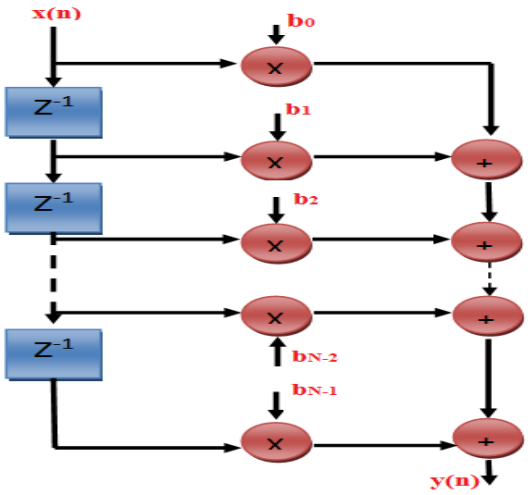

Figure 1: Direct realisation of FIR filter

With the rapid development of digital signal processing, FPGA platforms have been improved in term of reconfigurable hardware design for high performance processing [5]. It is widely accepted that the FPGA can be reprogrammed according to designer needs by using Very high speed integrated circuits Hardware Description Language (VHDL).There are several applications of using FPGA such as, speech recognition, audio processing, high performance computing and Image Processing[6,7]. It has been mentioned that the International Technology Roadmap for Semiconductors (ITRS) aims to make the objects smaller and keep the cost lower [8]. For example, the FPGA has many components and not that expensive. This means that the size of the circuit components has been reduced and in the same time, the performance has been increased. In this case, the power consumption of the electronic devices generally can be decreased.

By comparing with Application Specific Integrated Circuits (ASICs), where the device can be built for the specific design, FPGAs can be programmed multiple times with desired application. In thiswork, cyclone II FPGA is used which has many features. For instance, it is the lowest cost FPGA series, Embedded DSP

multipliers and embedded processor support [9]. In addition, the Altera DE1 board has the cyclone II FPGA and many other features as shown in Figure 2. This board is suitable for the development of complicated digital systems, as well as for an assortment of design projects.

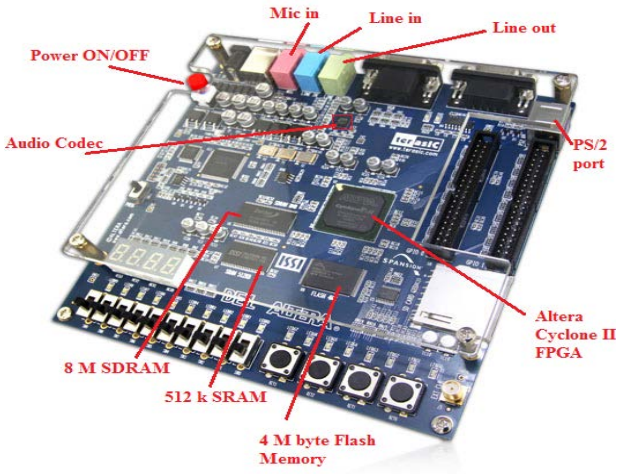

Figure 2: Altera DE1 board

2. Design Specifications and implementation

Three blocks are designed and analyzed in order to understand how the whole system is operated. These blocks are illustrated in Figure 3.

1. Codec initialization block.
2. S2P Adaptor (serial-to-parallel and parallel-to- serial).
3. FIR filter block

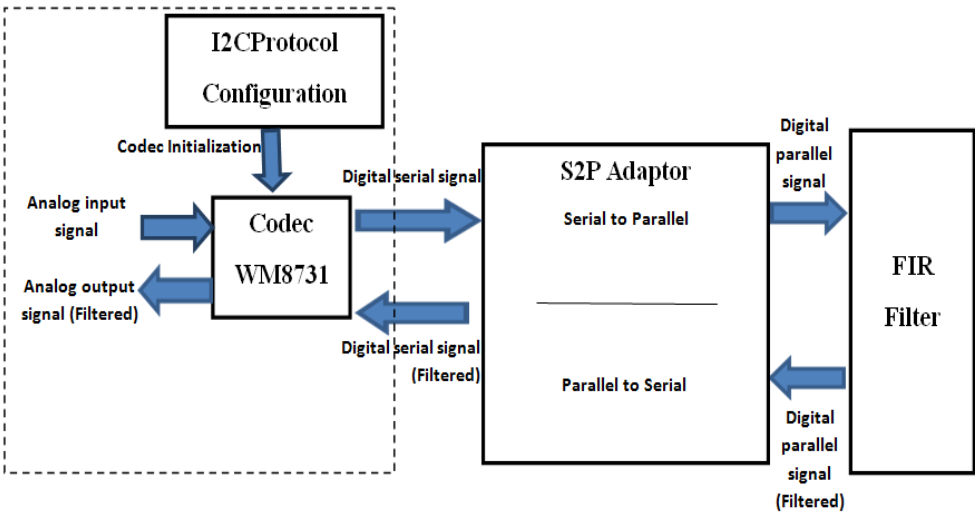

Figure 3: The audio digital filter system

In order to obtain a filtered audio signal, there are some stages which need to be completed. Firstly, the initialisation of the Codec block, which depends on I2C protocol because the Codec does not work in any particular configuration when the power turns on. Secondly, in theS2P Adaptor block, the serial digital signal is converted to parallel digital signal and is transmitted to the FIR filter. Finally, the filtered signal needs to be converted again to serial signal through the same block of S2P and send it back to the Codec output port.

2.1. Codec initialization

The initialization stage is very important to configure the work of the Codec with some features. For example, the Codec has a sample rate 44100 Hz and two input channels, which are right and left channel. In this design the left channel only has been chosen. These options controlled by data bits which are stored in registers. When these data bytes loaded into the registers, which have address and data, the actual data of the audio signal will be communicated through the control interface specially the pins SCLK and SDIN[10].SDIN is used for serial data and SCLK is used for clock serial data. In order to achieve the initialisation of the Codec, I2C protocol is used. In practice, the concept of this protocol is to determine the start and the end condition of the SDIN signal and then control the transmitted data which contains the configurations. For instance, after sending each eight bits the Codec should receive an acknowledge bit (ACK) from the FPGA to confirm that the data has been received correctly as illustrated in Figure4.

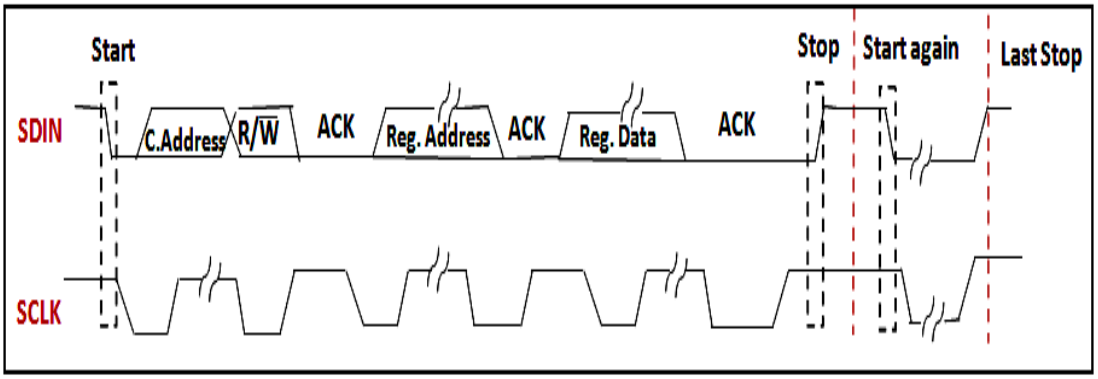

Figure 4: I2C Protocol

SDIN contains an important input data to initialise the Codec and these data consist of 11words and each word has 24 bits. In particular, each transmitting word start with 8 bits, which can be divided into 7 bits of the chip address (Codec) and one bit R/W to specify either read or write option. Another8 bits consider the register's address. The final8 transmitted bits that contain the actual data to configure the Codec. All these transmitted data are controlled by SCLK which works in maximum frequency500 kHz, however; it can work lower than this number to verify its work. Figure 4 demonstrates that the transmitted data should be started by detecting the start condition. The start condition can be represented as SDIN changes from '1' (high) to '0' (low) which is falling edge, while the SCLK stays high. After this processing the data will transmit starting from MSB to LSB. If a wrong address is received, the Codec would wait for the new start condition. When the address bits are correct and a R/W bit is 0 then SDIN sends 8 bits and waiting for (ACK). This process will be repeated for each word

(24 bits) and after receiving the last ACK, the stop condition will be detected. This condition occurs by detecting the rising edge of SDIN. In other words, the SCLK signal stays high and SDIN goes from low to high. After sending all data configurations, the Codec will back to the idle condition and wait for another start condition.

In order to understand the all 11 data words, it is important to know the content of each register. That means the I2C Protocol has different options which can provide the designers the opportunity to achieve what they need. It is vital to understand all various options with the details of all the registers, as shown in Table 1.

Table 1: Register Map Description

REGISTER ADDRESS	REGISTER DATA	DESCRIPTAION
R0 (00h)	000011111	Set the Mute and the Volume of Left Line channel.
R1 (01h)	000110111	Set the Mute and the Volume of Right Line channel.
R2 (02h)	001111001	Set the output volume and control the Left headphone out.
R3 (03h)	000110000	Set the output volume and control the Right headphone out.
R4 (04h)	011010010	Analogue Audio Path Control and set (ADC function).
R5 (05h)	000000001	Digital Audio Path Control
R6 (06h)	001100010	Power Down Control, this register can enable or disable the power down inside the Codec.
R7 (07h)	001000011	Digital Audio Interface Format. The input data indicate Codec's mode and the data input length (16 bits).
R8 (08h)	000100000	Sampling control. For example the sampling frequency is 44.1 kHz.
R9 (09h)	000000001	This register controls the active interface.
R15 (0fh)	000000000	Reset the device.

2.2. S2P Adaptor block

The S2P adaptor block converts the serial digital data from the Codec to parallel data and pass it to the FIR filter block. Then it receives the processed data from the FIR block after converting it from parallel to serial as shown in Figure 5. In addition, it has two interfaces: digital audio interface and paralleling interface .

In the digital audio interface, BCLK can be defined as a clock signal which is generated by the FPGA to distinguish the first bit of the data until reach the final bit (16 bits). This is distinguished by BCLK, when it goes from high to low (falling edge) and LRC signal is high. However, in this design the reading of the first bit happens in the middle of the bit because it is reasonably stable. In other words, the reading process (transmitting) occurs when BCLK goes from low to high (rising edge), while the writing process (receiving)

occurs on the falling edge of BCLK as demonstrated in Figure 6.

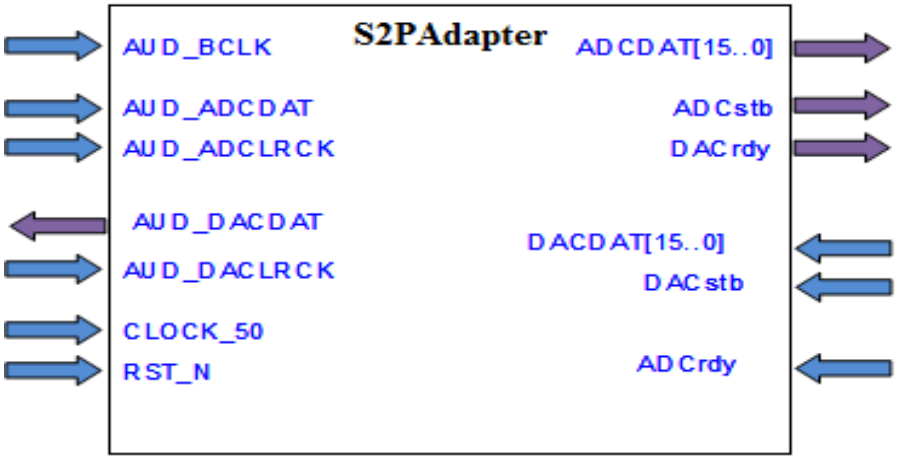

Figure 5: Block diagram of S2P Adapter

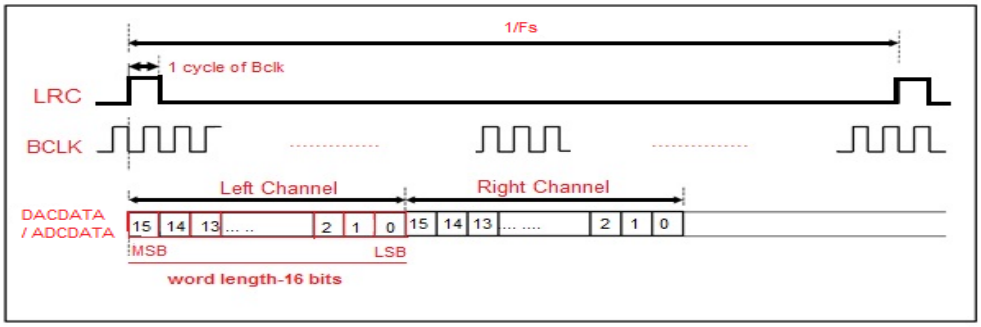

Figure 6: Digital audio interface

From Figure 6,DACDAT is audio output serial data, while ADCDAT is an audio input serial data. LRC signal indicates the beginning of ADCDAT and DACDAT data and it works as a reference with BCLK. Furthermore, the input data (ADCDAT) and the output data (DACDAT) begin to transmit simultaneously starting with the MSB. Each bit takes one cycle of BCLK during transmit or receive the data of the left channel and right channel. There are idle time intervals (32 bits) which can be used as security to give FIR filter time to process the data. This period of time starts after the left and right channel finished until the next LRC.

Parallel interface is very important to synchronise the signals between the sender and the receiver to make both blocks work properly. In order to achieve this aim, handshaking technique is used. This technique depends on two signals which are Strobe and Ready as illustrated in Figure 7.Handshaking technique is a simple protocol between S2P and FIR blocks. In particular, Strobe signal indicates that the packet of data (16 bits) is available and ready to send while Ready signal indicates whether the receiver is busy or idle to receive this packet of data. For example, when S2P Adapter has a packet of data ready to send to the FIR, the STBin signal will set to be high. At this time, if the RDYin signal is low (generated by FIR), which meansthere is agreement to communicate between both blocks. Whereas, if RDYin is high, which means FIR filter busy at this moment,

wait until finished processing and then RDYin will set to be low by FIR. The same protocol exactly repeats when the FIR filter block ready to send the data back to S2P Adapter.

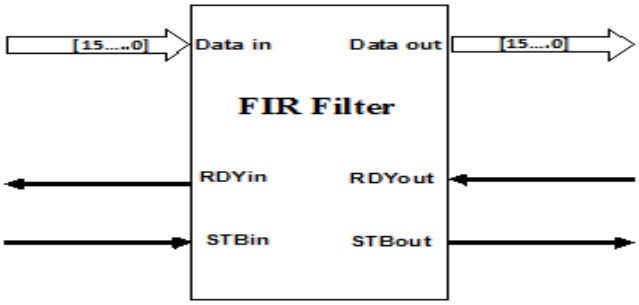

Figure 7: Strobe/Ready handshaking (parallel interface)

2.3. FIR filter block

There are several specifications on FIR filter block, particularly the number of coefficients, taps, and data shifter stages. The number of coefficients and taps are eight, and the coefficients have the following values: **-1260, 7827, 12471, 16384, 16384, 12471, 7827, -1260.** Hence, the complete equation of the FIR filter can be written as:

y(n)= -1260 x(n) + 7827 x(n- 1) + 12471 x(n- 2) + 16384 x(n- 3) + 16384 x(n- 4) + 12471zx(n- 5) + 7827 x(n- 6) + -1260 x(n- 7).

It can be observed from the equation above that the data shifter of the FIR filter has 7 stages; each stage is a 16 bit register. In addition, one Multiplier is required to multiply the output of each 16 bit register from the data shifter by the coefficients of the FIR as shown in Figure 8. Another requirement is an accumulator, which might need 35 bits to add the results from the multipliers and send the final result, which contains the most significant 16 bit, to the next stage.

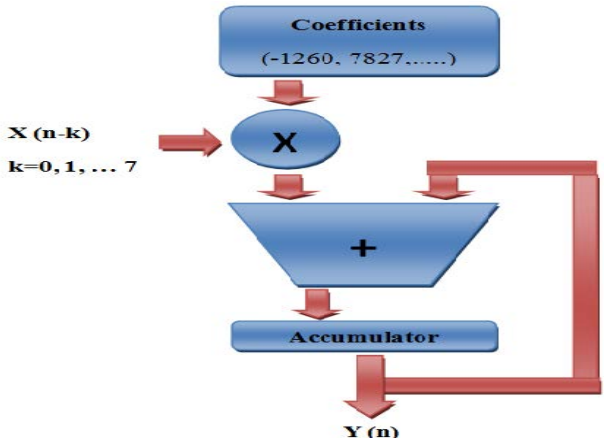

Figure 8: The concept of FIR design

The frequency response of this FIR filter is given in figure 9.

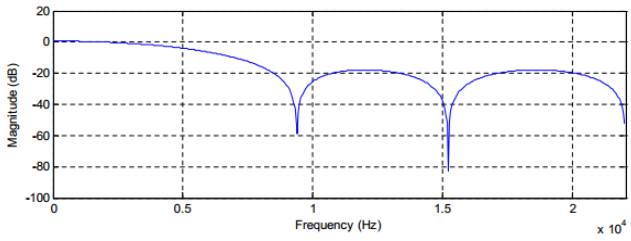

Figure 9: The frequency response of the FIR filter

According to the frequency response, the filter behaves as low pass. This means that the input signal will pass through this filter until reach the stop band area which starts approximately at 9 kHz. Finally, after designing the filter based on its specifications, and designing the other blocks that they essential for the whole system. All these blocks are implemented on the FPGA by writing a code of each block in VHDL language .Then, these codes are simulated with the Quartus II software tool.The correct results are obtained after using the correct test bench in the Quartus II.

3. Results and Discussion

3.1. Codec initialization Results

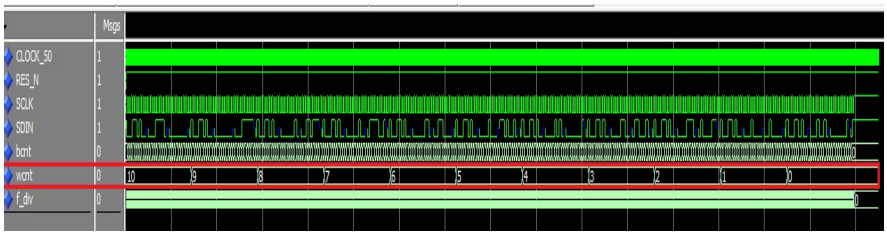

Figure 10: Final simulation of the Codec

I2C protocol configuration was implemented to initialise the Codec. This achieved by loading SDIN with eleven words of data register as shown in Figure10. As mentioned in the specification of the Codec that each word have 24 bits with 3 acknowledgement bits and 2 bits for start and stop condition. Therefore, the total number of transmitting words is 29 bits. In order to prove this specification, Figure 11 is provided.

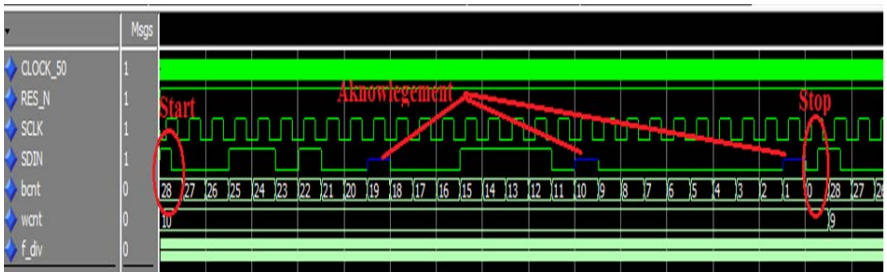

Figure 11: I2C protocol

From Figure11, it can be observed the start bit (bcnt=28), the first word (24 bits) of SDIN is started at bcnt =27 (00110100 Ack 00011110 Ack 00000000 Ack). Furthermore, after sending each group of 8 bits, the Codec receives an acknowledgement bit (Ack) from the FPGA to confirm that the data received properly.Finally, the stop bit takes place at (bcnt=0) to indicates that the Codec work properly.

It is noticed that the blue line is the high impedance 'Z' and it is very helpful for debugging to recognise the right place of the acknowledgement. When SDIN is set to '0', it is difficult to observe the acknowledgement among the data around, especially if shifting is happening.

3.2. S2P Adapter block Results

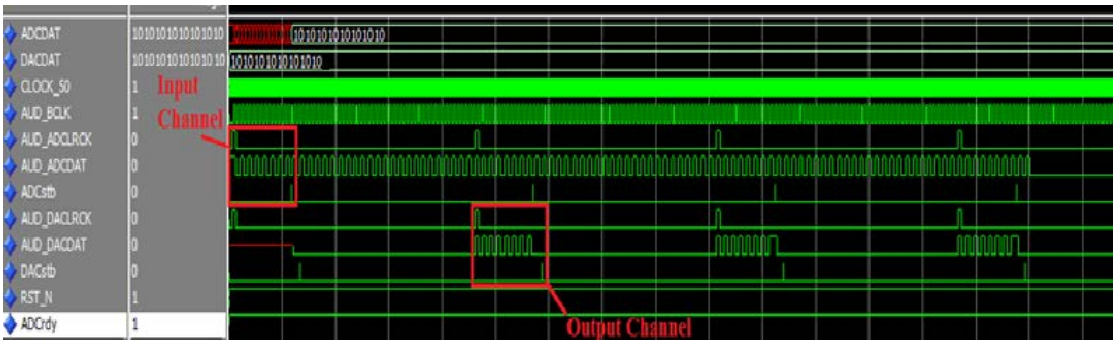

Figure 12: Input and output channels of S2P Adapter

In S2P adaptor block, there are two channels: Input channel and Output channel. Figure 12 depicts the simulation for both channels. In order to check the principle of each channel separately, Figure 13 and 14 are provided.

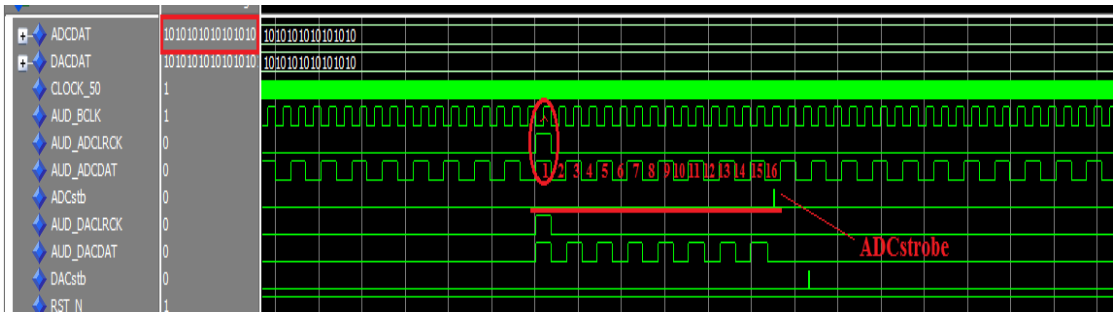

Figure 13: Input channel protocol

When AUD_ADCLRCK goes high and AUD_BCLK goes from low to high (rising edge), the reading process takes place in the middle of the bit as displayed in Figure 14. It is shown that after 16 clock cycles of AUD_BCLK, the input data transfers successfully to ADCDAT and then ADCstb is set 1 for one clock cycle of the main clock (50 MHz) by S2P Adapter. Which indicates that the input channel (serial to parallel) works correctly.

In the output channel, the same protocol was used but in this case the S2P reads the parallel data from the FIR block at the falling edge of AUD_BCLK. S2P Adapter then converted the data to serial inside DACDAT, which has "1010101010101010". FIR sets a strobe high to indicate the end of processing. Therefore, the output channel works properly as shown in Figure 14.

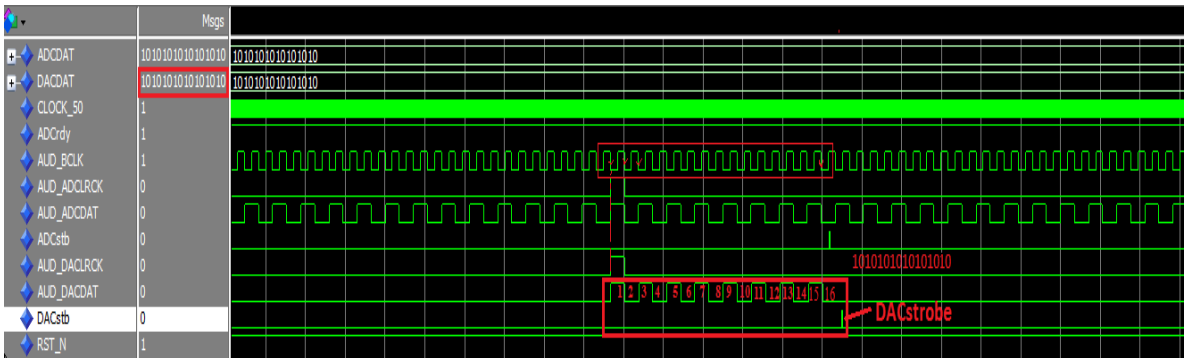

Figure 14: Input channel protocol

3.3. FIR filter Results

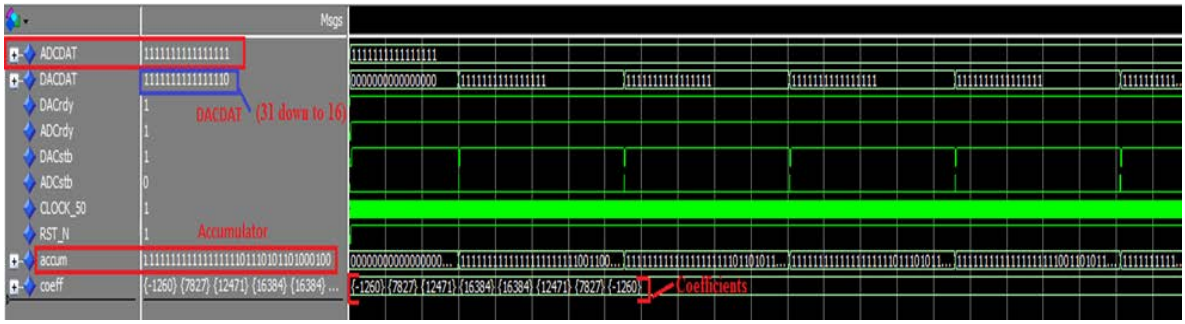

Figure 15: Final simulation of FIR filter

The FIR filter is very important block in this design, which can be considered as the core of the system. Hence testing this block is essential in order to obtain the final aim of this project. Figure 15 demonstrates the correct result of FIR filter.

Particularly, the input data (ADCDAT) are chosen to be"1111111111111111" (-1 in decimal) to make the test easy. Because the value of (-1) does not change with a shifting process due to the fact that all the bits are '1'. Theoretically, if the FIR filter works correctly, the output must be the addition of all the coefficients: y(n) = -1*(-1260 + 7827 + 12471 + 16384 + 16384 + 12471 + 7827 + -1260)=-70844.

Therefore, it can be observed that the decimal value of the accumulator is (-70844). This is equivalent to the binary number"11111111111111110001010010111100" which is highlighted in Figure 17. According to the specification, the DACDAT will take the bits (31 down to 16) from the 'accum'. So it will have "1111111111111110". This indicates that FIR filter works properly.

3.4. Hardware test

In The input signal is generated as a sine wave with different frequencies by using a signal generator. Moreover, an oscilloscope is used to obtain the final figures. Practically, in order to test the FIR filter in real environment, the input signal is connected to the DE1 board through LINEIN port. The output port (LINEOUT) is connected to an oscilloscope and through changing the frequency of the input signal, the frequency response of the FIR filter can be noticed as shown in the following figures.

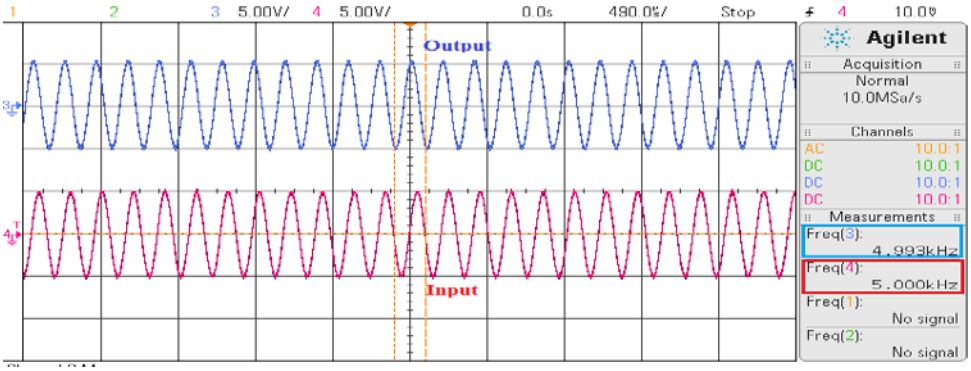

Figure 16: Input and output at 5 kHz

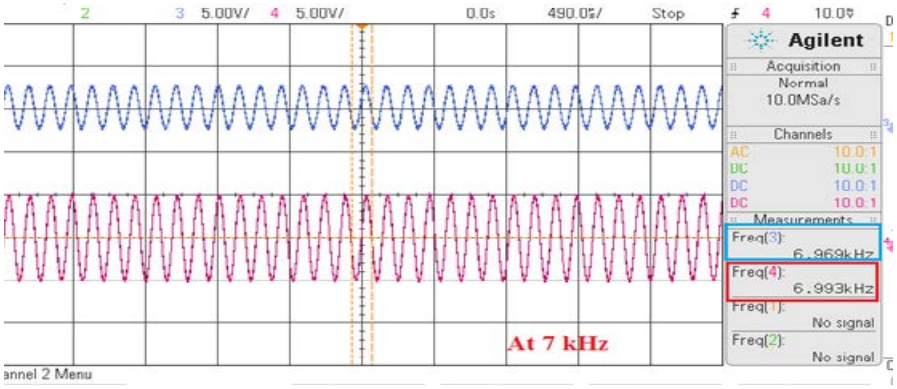

Figure 17: Input and output at 7 kHz

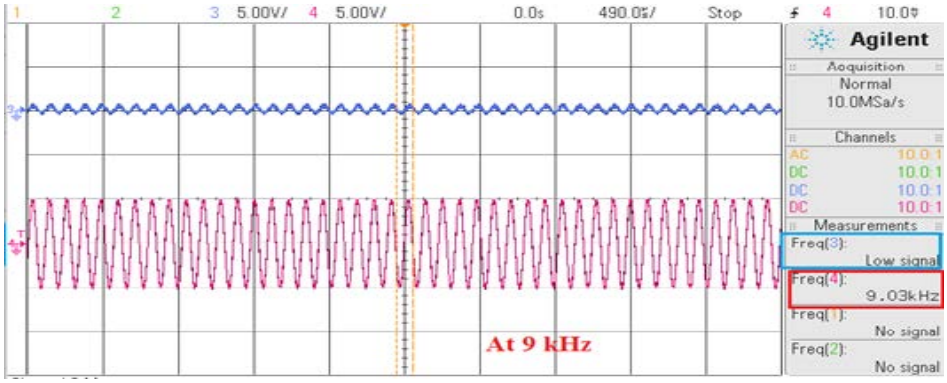

Figure 18: Input and output at 9 kHz

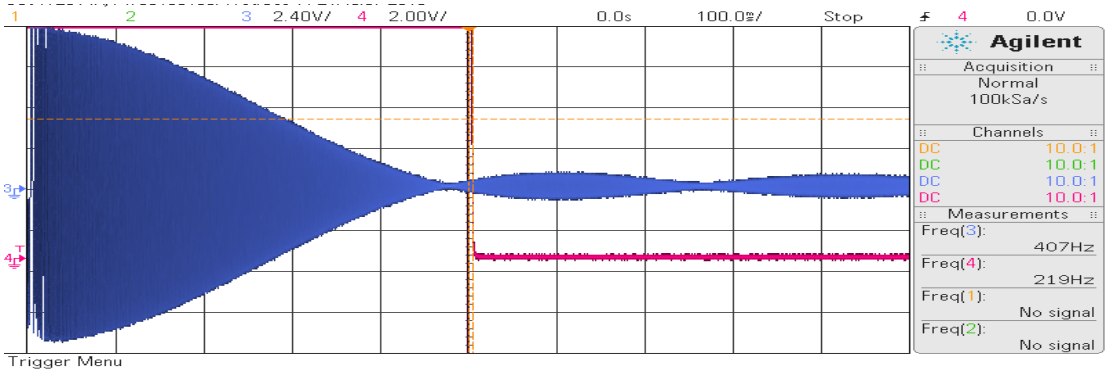

Figure 19: the output of sweep sine wave (100 Hz to 20k)

When the frequency of the input signal is increasing, this leads to decrease the amplitude of the output signal. From Figure 17, it can be seen that the amplitude of the output signal at7 kHz was decreased. In Fig 18, the amplitude of the output signal is decreased to be small value at 9 kHz because of the behaviour of the low pass filter.

In order to observe all the range of audio signal frequencies, the sweep sine is used as an input signal with start frequency of 100 Hz and stop frequency of 20 kHz. Therefore, it is easy to notice the frequency response of the FIR low pass filter as demonstrated in Figure19.

4. Conclusion

In conclusion, it is found that the implementation of the audio FIR filter on FPGA is possible. VHDL used to implement the FIR filter due to the powerful features of this language to divide the full design into sub designs. In Codec initialization block, I2C protocol is used to control the transmitted data, which contains the necessary configurations. This was implemented by detecting the start bit then sent the first word (24 bits) until the stop bit detected. This procedure repeated eleven times due to the number of data words that are used. In addition, after each 8 bits of the transmitted data, the Codec received an acknowledgement to confirm that the FPGA has received the configurations data correctly.

S2P Adapter block has two channels: input channel and output channel. The input channel is used to convert the serial digital data to parallel digital data and pass it to the FIR filter block. The data were 16 bits, which represent the left channel of the Codec. This is achieved by sending the MSB (15) first then the rest of the bits. While the output channel has converted the parallel data from FIR block to serial data again. The third block is FIR filter. This block was implemented by using shifter and accumulator to obtain the final filtered audio signal. After multiplying each output of the shifter (16 bits) by its corresponding coefficient, the results added together inside the accumulator. Finally, the FIR low pass filter sent the 16 bits of accumulated data (31 down to 16) to S2P Adapter block.

Furthermore, the main three blocks combined together in order to implement the whole work. Other blocks such as Codec clock block and Synchronous blocks are added to the full design to make it work properly. These

additional blocks are available in the library. The last step was to debug the final design from errors and then download it on DE1 board. In the hardware test, a sine wave was used as input with different frequencies. So when the frequency is increasing the output affected due to the behaviour of the low pass filter. It is recommended that writing a test bench is essential to verify whether any program work properly or not. It is possible to design and implementation of a low-pass, high-pass and a hand-pass FIR Filter in the same FPGA due to its felxible features.

References

[1] Ruan, A.W., Liao, Y.B. and Li, J.X. (2009) An ALU-Based Universal Architecture for FIR Filters. IEEE Proceedings of International Conference on communications, Circuits and Systems, Milpitas, July 2009, 1070-1073.

[2] Panayotatos, P. (2005) Frequency Response of Filters. Rutgers University, New Brunswick.

[3] Manal, H.J. and Asaad, H.S. (2013) High-Pass Digital Filter Implementation Using FPGA. IEEE International Journal of Advanced Computer Science and Applications (IJACSA), 13, 41-50.

[4] Quayyum, A. and Mazher, M. (2012) Design of Programmable, Efficient Finite Impulse Response Filter Based on Distributive Arithmetic Algorithm. International Journal of Information Technology and Electrical Engineering, **1**, 19-24.

[5] Monmasson, E., et al. (2011) FPGAs in Industrial Control Applications. IEEE Transactions on Industrial informatics, 7, 224-243.

[6] Wenjing, H., Guoyun, Z. and Waiyun, L. (2011) Self-Programmable Multipurpose Digital Filter Design Based on FPGA. IEEE Proceedings of International Conference on Internet Technology and Applications (iTAP), Wuhan, August 2011, 1-5.

[7] Fakharian, Ahmad, Saeed Badr, and Mohsen Abdi. "Implementation of a frequency FIR filter as 2D-FIR filter based on FPGA." AI & Robotics (IRANOPEN), 2015. IEEE, 2015.

[8] International Technology Roadmap for Semiconductors (2009) Design. http://www.itrs.net/Links/2009ITRS/2009Chapters_2009Tables/2009_Design.pdf

[9] Altera Corporation (2008) Cyclone 2 FPGA Data sheet. Available at: http://www.altera.com/literature/ds/ds_cyc.pdf

[10] Wolfson microelectronics (2012) WM9731/WM9731L. Available at: http://pdf1.alldatasheet.com/datasheet-pdf/view/174586/WOLFSON/WM8731L.html