Real-time 3D sound realization

A Project Report

submitted by

Bilal Ali(15EC112)

Sathwik G S (15EC116)

Barun Kumar Acharya (15EC209)

under the guidance of

Dr. Sumam David

in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL, MANGALORE - 575025

October 19, 2020

TABLE OF CONTENTS

ABSTRACT **6**

1 Introduction **7**

1.1 Problem definition 7

1.2 Previous work 7

1.3 Motivation 8

1.4 Overview 8

2 Background and Implementation **10**

2.1 How do we hear 3D sound with only two ears? 10

2.1.1 Binaural Cues 10

2.1.2 HRTF 11

2.2 HRTF Database 11

2.3 The setup 12

2.3.1 MATLAB 13

2.3.2 Digital design - Version 1.0 13

2.3.3 Digital design - Version 2.0 18

2.4 FPGA Implementation 22

2.4.1 Input 22

2.4.2 Importing Design 25

2.4.3 Output 25

3 Testing and Observations **27**

3.1 Observation 27

3.1.1 Angle resolution 27

3.1.2 Panning 27

3.1.3 Testing Design 28

4 Conclusions **29**

4.1 HRTFs 29

4.2 Analysis of MATLAB implementation 29

4.3 Analysis of RTL Design 29

4.3.1 Analysis of Design - Version 1.0 30

4.3.2 Analysis of Design - Version 2.0 30

4.4 Final Inference 31

4.5 Future Work 31

LIST OF FIGURES

2.1 HRTF for source to the left 11

2.2 HRTF for source to the right 11

2.3 Visualization of points at which HRIR from CIPIC are recorded. 12

2.4 Overall Design 14

2.5 MAC Unit 15

2.6 XCORR Module 15

2.7 Correlation operation - at each iteration the corresponding elements are multiplied and added 16

2.8 Convolution Module 17

2.9 Maximum Correlation index to Filter Index Mapping 17

2.10 Modified XCORR 18

2.11 Dataflow for cross-correlation 19

2.12 Choosing filter 20

2.13 Comparison 20

2.14 Filter change at T=1s 21

2.15 Filter Coefficient Update 21

2.16 Transition between filters 22

2.17 Fade In/Out Generator 22

2.18 XADC Interfacing 23

2.19 XADC Register Interface 24

2.20 Detailed timing of events 24

2.21 Block Representation 25

2.22 Setup 25

2.23 I2S Protocol 26

2.24 Clock speed of different signals involved in I2S transmission 26

3.1 Input: Sine wave of 1kHz 27

3.2 Output: After panning 28

3.3 Resource Utilization 28

ABSTRACT

3D Audio is a generic name that encompasses ways to render audio like in reality and in particular its spatial dimension; it is the capability to produce and perceive sounds in any direction and at any distance. Localization of sounds in physical space plays a very important role in multiple audio-related disciplines.

Binaural recording is the most commonly used method to provide an immersive sound experience by means of headphone reproduction.The method is based on a time-frequency analysis of the spatial properties of the sound picked up by a simple dual microphone array, assuming source sparseness. Usually , in order to synthesize 3D sound effects using binaural systems, we use Head-related transfer functions (HRTFs) to describe the spectral filtering that occurs between a source sound and the listeners eardrum. Because of the characteristics of HRTF some three-dimensional effects in the area of a cone of confusion between front and back directions can be declined. In order to design the system , we derive a large set of HRTFs for different azimuth and elevations from already present databases and with the help other binaural cues reconstruct the signal to generate the appropriate effect. In this project we implement a system for 3D sound localization which synthesizes a signal in real-time and aim to achieve with minimal computational complexity. The output sound quality is evaluated by a subjective listening test.

CHAPTER 1
Introduction

1.1 Problem definition

We are trying to design a system that will enable the user to realize and virtually locate a sound source heard through headphones. However, the main criteria of this system would be to respond in real-time. This has become a hot topic in the past few years due to its expanding application in a lot of fields like games, home theaters and human aid systems. It can improve our experience in some interactive application together with haptic feedback.

1.2 Previous work

Initial research involved localizing the sound source rather than reproducing it. Several methods have been proposed in different transfer domains to fasten the process of localization.One of most robust methods to in time domain is given by the GCC-PHAT method as discussed by B. Van Den Broeck and A.Bertrand [1] derived from [2]. Moreover it has undergone modifications for more critical situations such as multiple sources, described in [3].
The amplitude panning method used in stereo and surround systems today, was invented in 1931 by Alan Blumlein. This method is designed to allow audio sources to be placed at different angular offsets whichs allows instruments to be placed in different positions to build up a sound scene utilising the sine/cosine panning law, or in more advanced cases in surround systems to place sources around a listener in the horizontal plane. The basic concepts 3D sound systems are explained in detail by E. M. Wenzel [4]. General information about the 3D localization of the sound can be realized on the basis of the HRTFs from the source to the right and left ears are discussed by Chong-Jin and Woon-Seng Gan [6], E, M.Wenzel [4]. The robustness analysis, localization characteristics, structural model for binaural sound synthesis and head related transfer functions are discussed in detail by C. Phillip Brown and Richard O. Duda [7] . HRTFs (head related transfer functions) allow localisation of sources in the vertical dimension.It was proposed by Wallach

[8] that dynamic head movements also facilitate the localisation due to head movements which produce variations in inter-aural differences to resolve the position of sources. Recent research by Ashby et al. (2011) investigated the influence of head rotation on the localisation of elevated sound sources.

In this work we used the HRTFs made available to public by the *Listen project*, which is a shared-cost RTD project in the Information Society Technologies (IST) Program of the European Commission's Fifth Framework Program. In the later stages we have also used HRTFs from the CIPIC Interface Laboratory.

1.3 Motivation

People hear sound in three dimensions and the perception of the spatial aspects of sound has been essential to human survival.Psycho-acoustic researchers have been studying this problem from the view of understanding the functioning of the human hearing system which makes sound localization possible in the real world, leading to the principles of binaural human hearing.

With advancements in the field of virtual reality, it becomes really important to make sure that along with visual effects ,the audio is also on the same page (3D effect). Most of the methods existing allow to reconstruct spatialized audio post-processing. We are trying to achieve the whole outcome in real-time. Along with solving the problem of timing constraint , we also want to minimize resource utilization, therefore designing an ASIC as such. In the end, we will try to get a standalone system which gives the user a live surround-audio like experience.

1.4 Overview

The setup includes a dual microphone arrangement which continuously records and the recorded sound is further filtered. This filtered sound is then played back over headphones. The selection of the filter is based on the location of the sound source w.r.t the center of the microphone setup. Filters are characterized, based on the azimuth and angle of elevation value .

In order to calculate the angle of sound source, we cross-correlate the signals observed on both the microphones and calculate the delay between them. Once this is calculated, appropriate filter is chosen and convolved with the recorded signals. Typically, sounds generated from headphones appear to originate from within the head. In the virtual auditory space, the headphones should be able to externalize the sound. Using the HRTF, sounds can be spatially positioned.

CHAPTER 2
Background and Implementation

2.1 How do we hear 3D sound with only two ears?

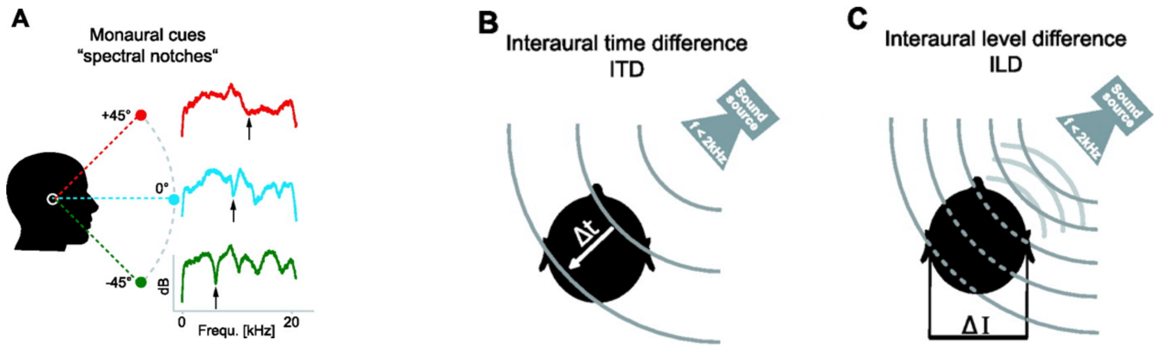
The auditory system uses the combination of 2 types of information to localize a sound: **Binaural Cues** (Time and Level) and **Head Related Transfer Functions** (HRTF).

2.1.1 Binaural Cues

When a sound source is located eccentrically, it is closer to one ear than the other and sound arrives later and weaker at one ear. Time and level difference perception between the two ears provide information about direction. These binaural cues are called Inter-aural Time Difference (ITD) and Inter-aural Level Difference (ILD). They are important for the horizontal localisation. The auditory system evaluates interaural time differences from: (a) Phase delays at low frequencies and (b) group delays at high frequencies. Inter-aural Intensity Difference (IID) or Interaural Level Difference (ILD) are highly frequency dependent and they increase with increasing frequency. Massive theoretical researches demonstrate that IID relates to the signal frequency f and the angular position of the acoustic source . The function of IID is given by:

$$IID = 1.0 + \frac{f}{1000}^{0.8} \times sin\theta \tag{2.1}$$

For frequencies below 1000 Hz, mainly ITDs are evaluated (phase delays), for frequencies



above 1500 Hz mainly IIDs are evaluated. Between 1000 Hz and 1500 Hz there is a transition zone, where both mechanisms play a role.

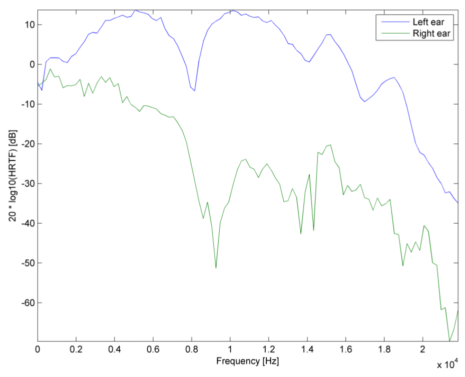


Figure 2.1: HRTF for source to the left

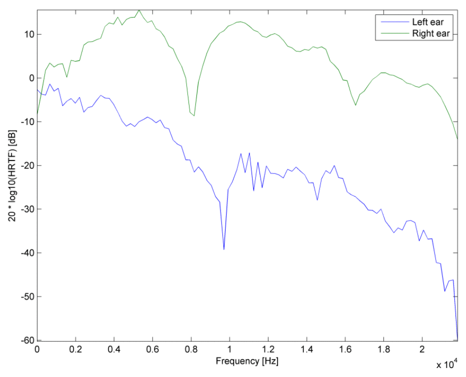


Figure 2.2: HRTF for source to the right

2.1.2 HRTF

Before reaching the entrance of the ear canal, a sound wave coming from one direction is transformed by the multiple reflections on our body, our face and more importantly our ears. This transformation, called Head Related Transfer Function (HRTF), is different for each direction of sound and has a specific spectral signature for each direction. It is a transfer function, describing how a sound from a specific point will arrive at the ear (generally at the outer end of the auditory canal). The brain has learned these signature over time and when it hears a sound with a specific signature, it can find out its direction in its HRTF memory. HRTF is important for vertical localization. A pair of HRTFs for two ears can be used to synthesize a binaural sound that seems to come from a particular point in space. HRTFs are typically measured in an anechoic chamber to minimize the influence of early reflections and reverberation on the measured response. HRTFs are measured at small increments of such as 15° or 30° in the horizontal plane, with interpolation used to synthesize HRTFs for arbitrary positions of . Even with small increments, however, interpolation can lead to front-back confusion, and optimizing the interpolation procedure is an active area of research.

2.2 HRTF Database

The area of 3D sound synthesis has been active since a very long time and many research labs have collected the HRIR (Head related impulse response) in well structured environment suitable for audio recordings. Some of these databases are from MIT, UC Davis (CIPIC database) and few privately funded labs.

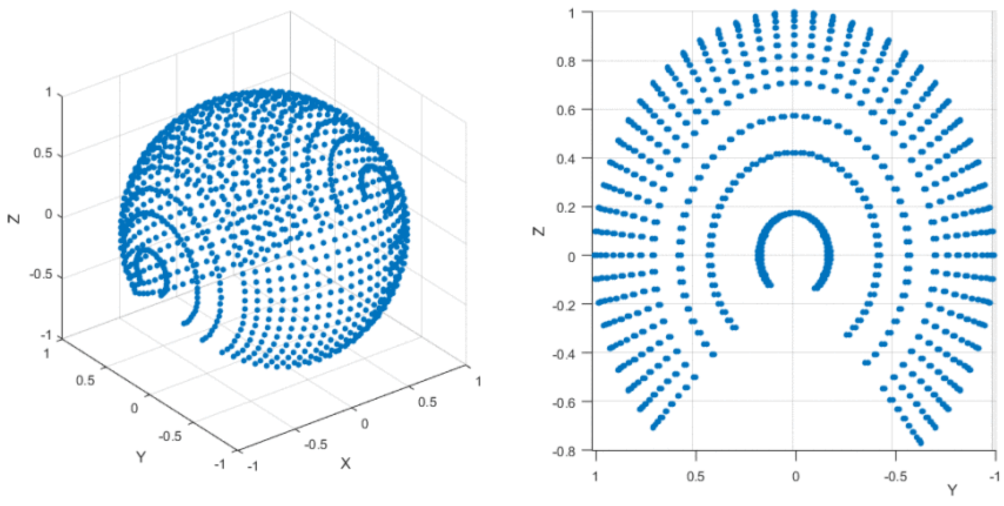


Figure 2.3: Visualization of points at which HRIR from CIPIC are recorded.

The database is maintained based on recordings from different subjects (with varying anthropometric data). Each subject is equipped with a pair of microphones and is made to sit in an anechoic chamber. Test sounds are played and the perceived signals on each ear is recorded. These sounds are played from different azimuthal as well as elevated angles. Some of the database includes anthropometric measurements for use in HRTF scaling studies.

2.3 The setup

The system we plan to design receives audio input , gets filtered and is played back through headphones with 3D-like effect.

For collecting audio samples, we recorded audio input using 2 microphones separated by 20cm (approximate distance between two ears). We used the National Instruments Data Acquisition module to simultaneously record the input from 2 microphones at 44.1 kHz sampling rate. The recorded data were passed as text files to MATLAB environment for further processing .This was as far as testing the unit was concerned. Later, during the actual test of the entire design we used MEMS Microphones (ADMP401) to acquire audio signals.

2.3.1 MATLAB

The hrtf database was loaded along with the audio file. Cross-correlation between input received at left microphone and the input received at the right microphone was performed. Based upon the results of cross-correlation , we get the time difference of arrival of audio input at the left microphone and the right microphone. Based on this time difference, we can predict the azimuthal and spatial information of the audio source.

$$\tau = \frac{Max(S_L * S_R) - Signallength}{SamplingFrequency} \tag{2.2}$$

where '*' operator denotes the cross-correlation operation on left and right microphone signals.

$$sin\theta = \frac{\tau c}{d} \tag{2.3}$$

where c is speed of sound in air , d is 20cm, θ is azimuthal angle .

Based on the above values, a particular HRTF filter was chosen and convolved with the input. The output we observed showed features clearly indicating spatial properties. In the same way we tested for different azimuthal angles using its corresponding HRTF filter coefficients. This experiment was done solely to understand and verify the functionality of the filters.

2.3.2 Digital design - Version 1.0

This section explains the first version of the digital design and implementation of the basic functionality of realizing 3D audio effect, which was done during the early stages of the project. After testing the following design and realising some improvements, changes were made and restructured in *Version 2.0* mentioned later in this section.

In order to reduce the complexity but maintaining the quality of filters , the filter coefficients were down-sampled from 512 (originally available in the database) to 64. The sampling rate of audio input, assumed to be 44.1 kHz.

The overall flow of data is as follows:-

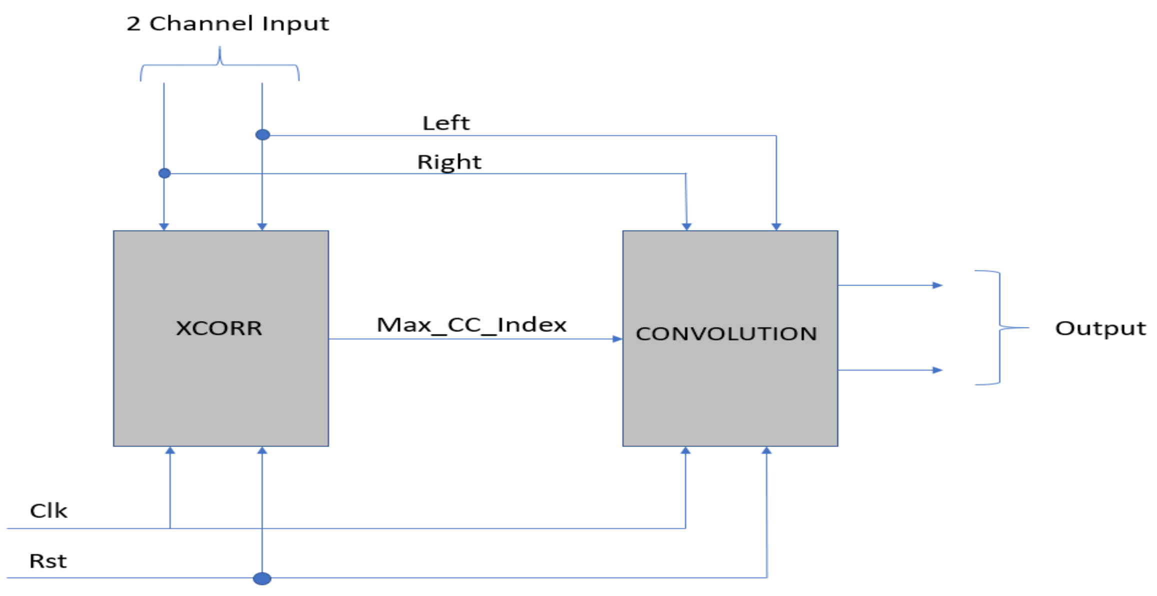


Figure 2.4: Overall Design

- 2 channel 16 bit input is given to the 'xcorr'(Cross Correlation) unit
- This unit collects 30 data samples and finds the delay between the two signals by cross-correlating them.
- Using this delay, input data samples are convolved with appropriate filters (indexed by the delay)
- The resultant 2 channel output is played back

Data format

The RTL design assumes all kinds of data to be in the following format :

- Inputs/Outputs must be signed 16 bits
- Each value should be in 1.15 format where MSB bit is the sign bit and remaining 15 bits are the fractional bits
- Range of values should be from -1 to 1

MAC Unit

This is the most important arithmetic unit of the overall design. The following Figure 2.5 shows the data flow. It uses only **One** multiplier (16 bits) and keeps accumulating the resultant 32 bit iteratively. The final result is stored in a register, Figure 2.5. One must be careful with the bitwidth of the result and truncate proper bits.

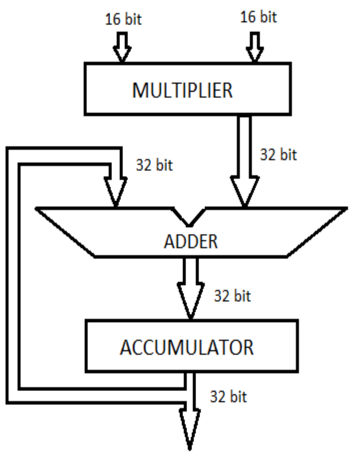


Figure 2.5: MAC Unit

Note: MAC output must truncated to 16 bits, else will lead to miscalculation in further stages.

XCORR

The main function of this module is to cross correlate between any two signals. Since the setup involves 20cm between the two recording devices and assuming speed of sound to be 343 m/s, the maximum delay possible comes out to be around **27** samples. Therefore, the signals correlated are of length 30 samples. Initially, the buffers of both the channels

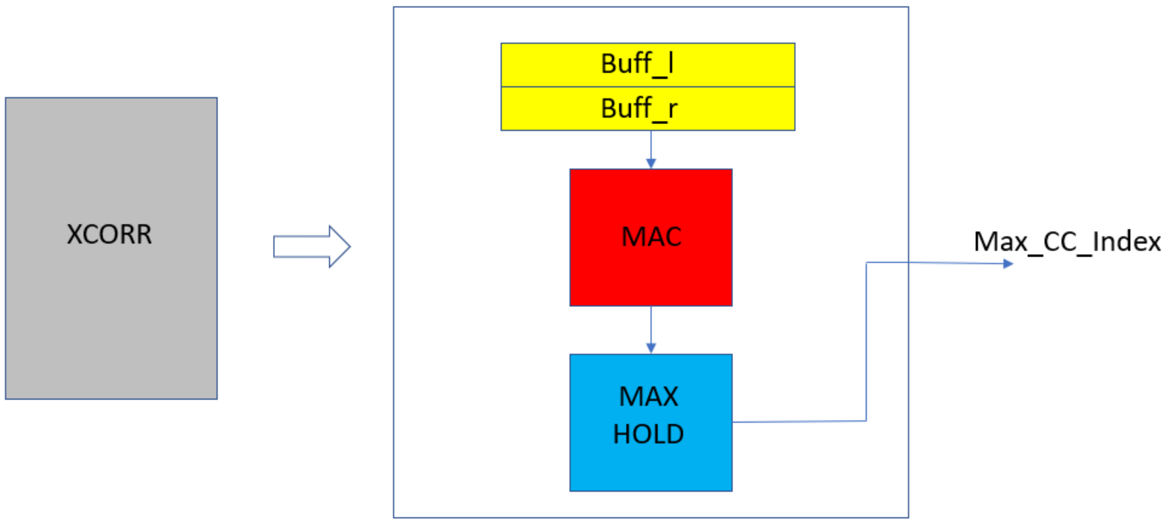


Figure 2.6: XCORR Module

collect those 30 samples and pass it on to the **MAC** unit. Here, the signals are multiplied and added. One of the buffer is shifted and the process is repeated until the buffer slides out, Figure 2.7. The 'Max-hold' unit keeps track of the maximum value of the MAC

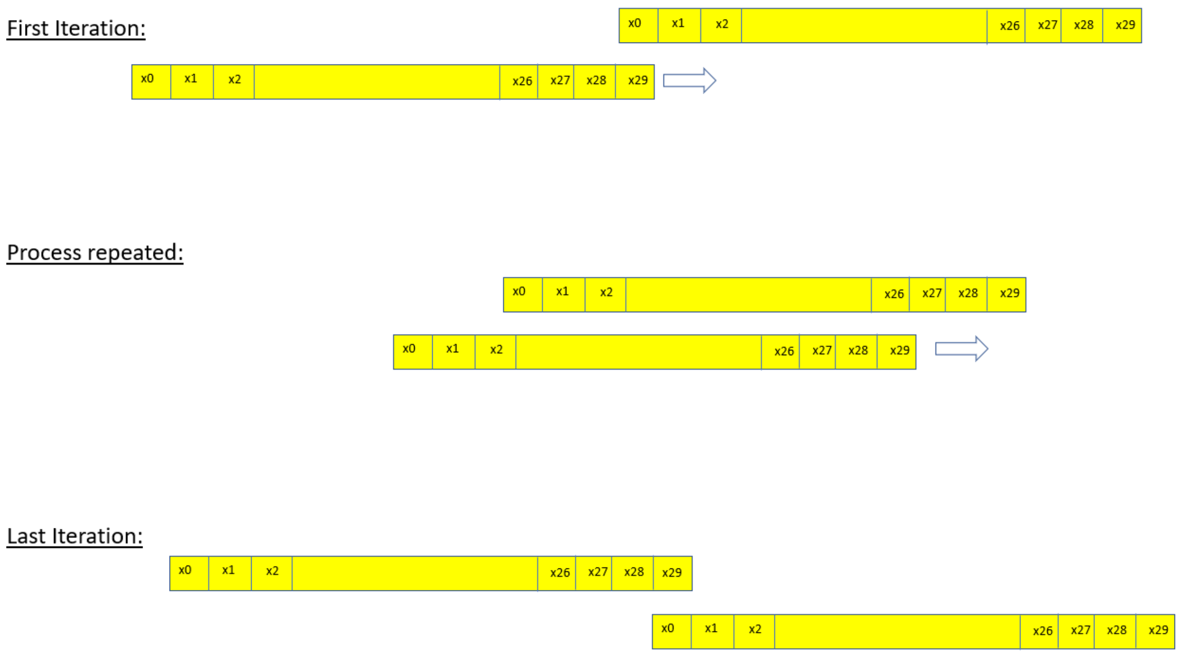


Figure 2.7: Correlation operation - at each iteration the corresponding elements are multiplied and added

unit. It updates whenever a new maximum is found. After completion of the process, the index of shift corresponding to maximum MAC value it passed as the output.

Calculation of clock speed required: Given that the sampling rate = 44.1 kHz, at the end of acquiring 30 samples within a sample period there must be 59 shifts (length of cross-correlation output) with 30 MAC operation in each shift. Thereby giving rise to clock requirement which is 59x30 = 1770 times faster than 44.1 kHz. This implies clock speed required is = 78 MHz.

Convolution and Filter Bank

This module performs 2 tasks; selecting appropriate filter from a filter bank and convolving the same with input data samples. The buffer lengths are once again 30, since these are the same samples whose cross-correlation is being computed. But, each filter corresponding to a certain azimuthal angle has **64** coefficients, therefore we would have to use **34** samples of previous frame.
In order to choose appropriate filter, the index from the **xcorr** module is mapped accordingly to filter index, see Figure 2.9.

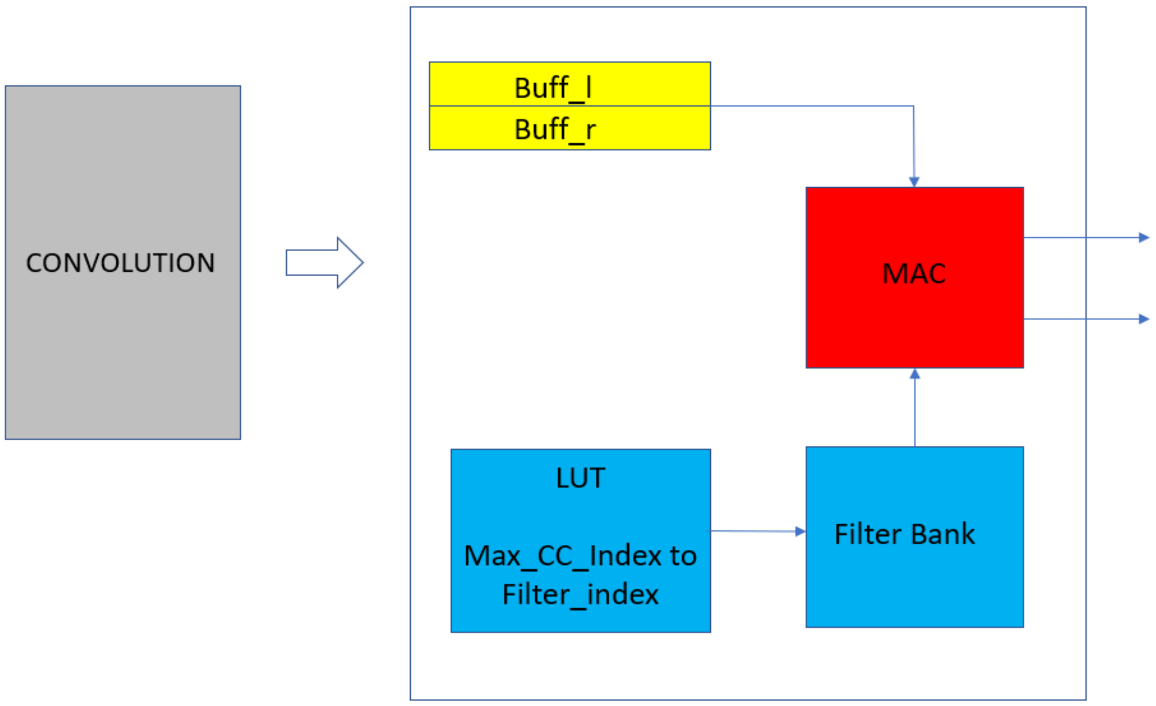
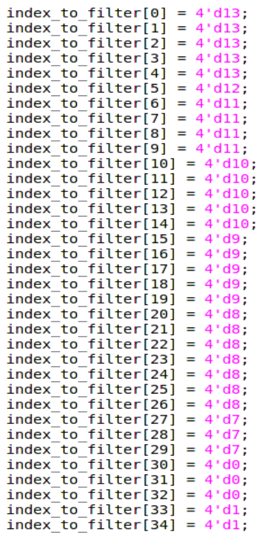


Figure 2.8: Convolution Module



```
index_to_filter[0] = 4'd13;
index_to_filter[1] = 4'd13;
index_to_filter[2] = 4'd13;
index_to_filter[3] = 4'd13;
index_to_filter[4] = 4'd13;
index_to_filter[5] = 4'd12;
index_to_filter[6] = 4'd11;
index_to_filter[7] = 4'd11;
index_to_filter[8] = 4'd11;
index_to_filter[9] = 4'd11;
index_to_filter[10] = 4'd10;
index_to_filter[11] = 4'd10;
index_to_filter[12] = 4'd10;
index_to_filter[13] = 4'd10;
index_to_filter[14] = 4'd10;
index_to_filter[15] = 4'd9;
index_to_filter[16] = 4'd9;
index_to_filter[17] = 4'd9;
index_to_filter[18] = 4'd9;
index_to_filter[19] = 4'd9;
index_to_filter[20] = 4'd8;
index_to_filter[21] = 4'd8;
index_to_filter[22] = 4'd8;
index_to_filter[23] = 4'd8;
index_to_filter[24] = 4'd8;
index_to_filter[25] = 4'd8;
index_to_filter[26] = 4'd8;
index_to_filter[27] = 4'd7;
index_to_filter[28] = 4'd7;
index_to_filter[29] = 4'd7;
index_to_filter[30] = 4'd0;
index_to_filter[31] = 4'd0;
index_to_filter[32] = 4'd0;
index_to_filter[33] = 4'd1;
index_to_filter[34] = 4'd1;
```

Figure 2.9: Maximum Correlation index to Filter Index Mapping

We observe that as the delay increases, even a small change in it causes change in filter selection. This implies that around midway the resolution is highest. After selecting the filter, the coefficients are then convolved with input samples (of length 30) + past 34 samples to give the output.

The filters used are derived from the database, of which filters corresponding to only 0° elevation are used here and there are **13 different** azimuthal angles each with 64 coefficients. All these coefficients are stored in banks of registers and indexed accordingly.

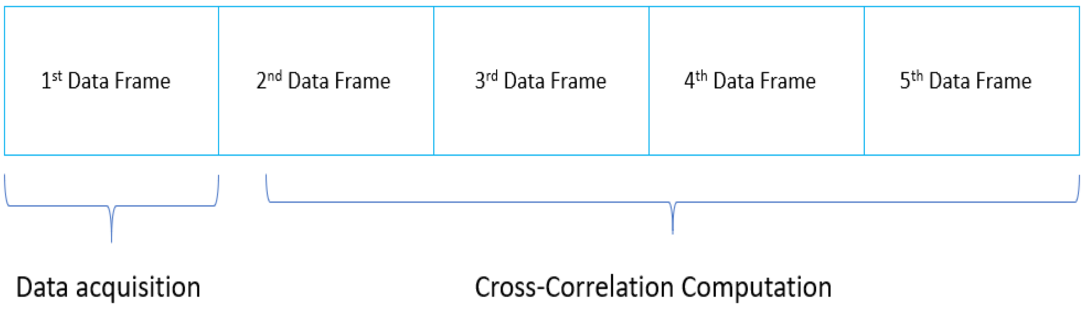


Figure 2.10: Modified XCORR

2.3.3 Digital design - Version 2.0

This section will include all kinds changes made after checking the output of the previous design in real time. The following problems were faced and needed to be addressed:

- Filters were changing too fast, resulting in not-so-pleasant output
- Given the output sample rate, the computational main clock speed required was too high
- Audible glitch during transition from one filter to another

The filters were then changed to 100 coefficients *(Initially 200 length – CIPIC HRTF Database)*. This database had more number of filters for different directions. In the azimuthal sense, it has 25 different filters. Although the MAC unit remains the same , changes were made in the **XCORR and Convolution unit**.

Modified XCORR

Previously, only 30 samples were used to correlate and this was done on every 30 sample data frame. This sometimes led to random angles due to possible presence of noise samples more than the actual audio/speech samples. Considering the sampling rate of 44.1 kHz, 30 samples = 0.68 ms which is extremely short, hence it is required to compute the cross-correlation at a slower rate with longer frames.

The modified XCORR unit now considers a data frame of 100 samples long illustrated in Figure 2.11. Unlike the previous design which **collected samples** and computed the output **before** the next data frame resulting in higher clock speed requirement, the new design now collects 100 samples and computes the output in the time period of next 400

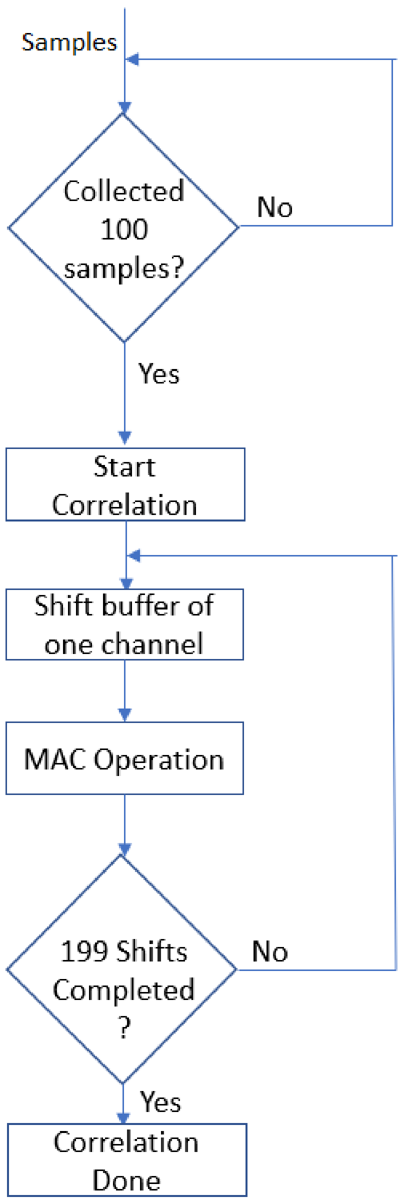


Figure 2.11: Dataflow for cross-correlation

samples as shown in Figure 2.10. In other words, cross correlation of **100 samples but of every 5th data frame**, thus computing period of **4 data frames** resulting in relaxed clock speed requirements. These changes will make sure that the rate of correlation is decreased and the clock speed required is also significantly **reduced**.

Modified Convolution unit

This unit will now deal with convolving 100 data samples with a corresponding filter. The changes to be made here should tackle the issue of audible glitch while transition between filters. Now that the cross-correlation module (XCORR) gives out new index every 5th frame (5*frame.lengths = 11.3 ms), it is important to make sure that the valid cross correlation index is used to choose the appropriate filter.

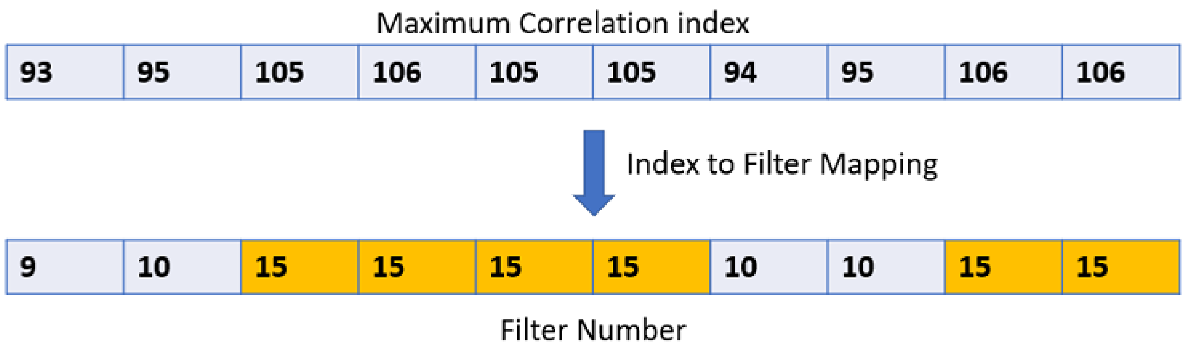
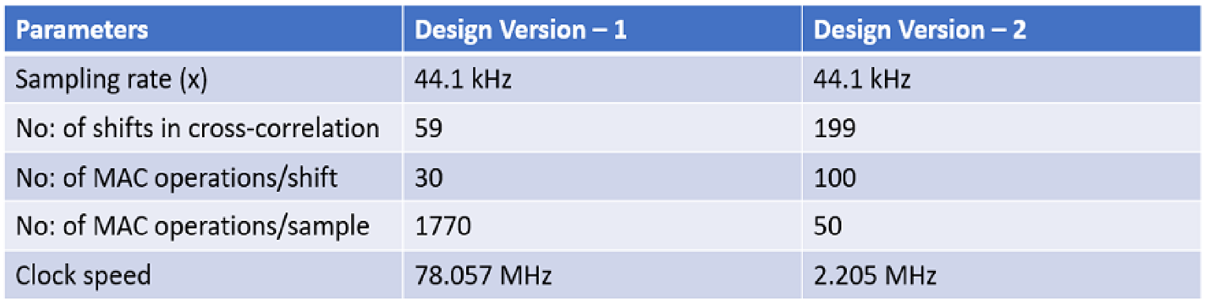


Figure 2.12: Choosing filter

In order to choose the valid filter, we then keep track of **past 10** cross correlation indices mapped to corresponding filter as shown in Figure 2.12. The filter which occurs the **most amount** of times is then chosen as the new filter (in this case shown filter 15 is chosen) and brought into the computation as shown in the following section. The index to filter mapping was redone to account for the new cross-correlation indices. This essentially means that there could be a change in filter in about **every 110 ms** which can still account for any change in location of the sound source.



Parameters	Design Version – 1	Design Version – 2
Sampling rate (x)	44.1 kHz	44.1 kHz
No: of shifts in cross-correlation	59	199
No: of MAC operations/shift	30	100
No: of MAC operations/sample	1770	50
Clock speed	78.057 MHz	2.205 MHz

Figure 2.13: Comparison

Live Convolution with time varying FIR filters

Amid the procedure of filter change, rather than totally supplanting the filter one should let the convolution with the present filter to get finished and yet begin convolving with the latest one. This process requires **2 convolution units** operating at the same time. There is another approach to accomplish the equivalent yet utilizing just 1 convolver [5]. During the exit phase of the present filter a portion of the initial coefficients don't add to the outcome, however has a job for the latest filter. In a way the filter buffer should be loaded up with **partial commitments** from both the filters, illustrated in Figure 2.15.

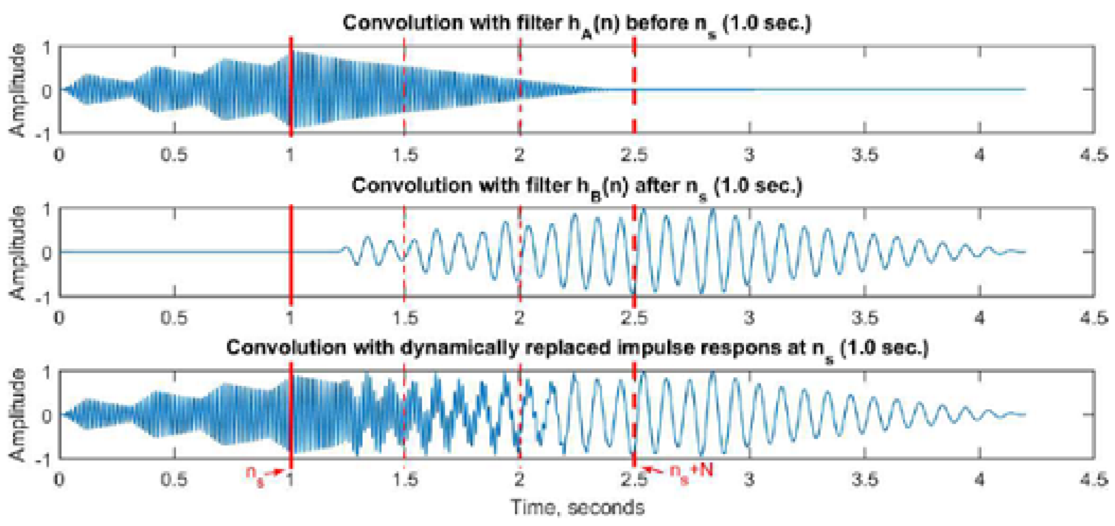


Figure 2.14: Filter change at T=1s

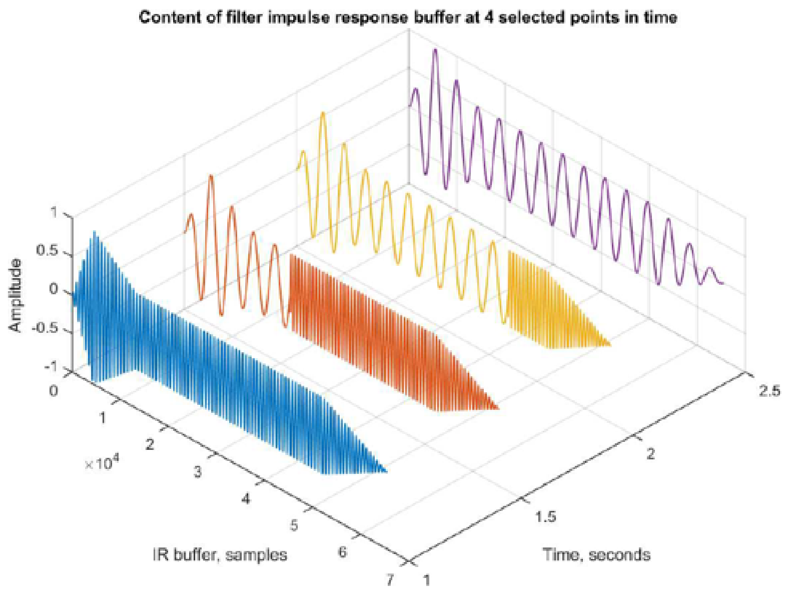


Figure 2.15: Filter Coefficient Update

For each sample update in the buffer to be convolved with, the coefficients are updated at the same time.

This approach is equivalent to the using 2 convolvers but the advantage being, requirement of only **1 multiplier unit** at a time.

Smooth Transition

To avoid audible glitches while transition we slowly introduce the new filtered output into picture rather than just abruptly appending it. This can be done by crossfading the output of a particular data frame with the previous filter and fade in the output with the

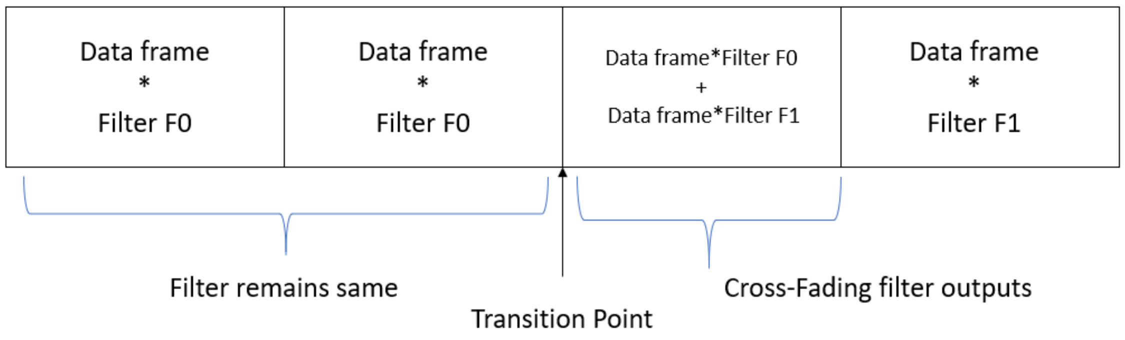


Figure 2.16: Transition between filters

current filter, as shown in Figure 2.16. Standard fading techniques such as linear, square-root, sin-cos etc can be used and **dynamically** generated without having to store them, as shown in Figure 2.17. Depending on fading in or fading out, values are incremented or decremented respectively. This value is then multiplied with corresponding filter outputs as shown in Figure 2.16.

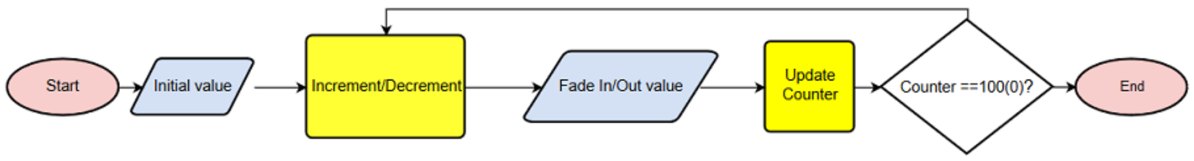


Figure 2.17: Fade In/Out Generator

All these changes resulted in overcoming the problems faced by the previous design.

2.4 FPGA Implementation

In order to test the design and check its functionality in real time **Nexys 4 DDR Artix -7 FPGA (now renamed as Nexys A7)** which has enough resources to accommodate our design requirements was used.

2.4.1 Input

The design requires a two channel input essentially coming from two microphones separated by about 20cm. Here, we have used **ADMP401 MEMS Microphones** which is omnidirectional, analog output, low power and has flat frequency response from 100Hz to

15 kHz suitable for near and far field applications. We must convert the incoming analog signals from the microphones to digital form using an **ADC**.

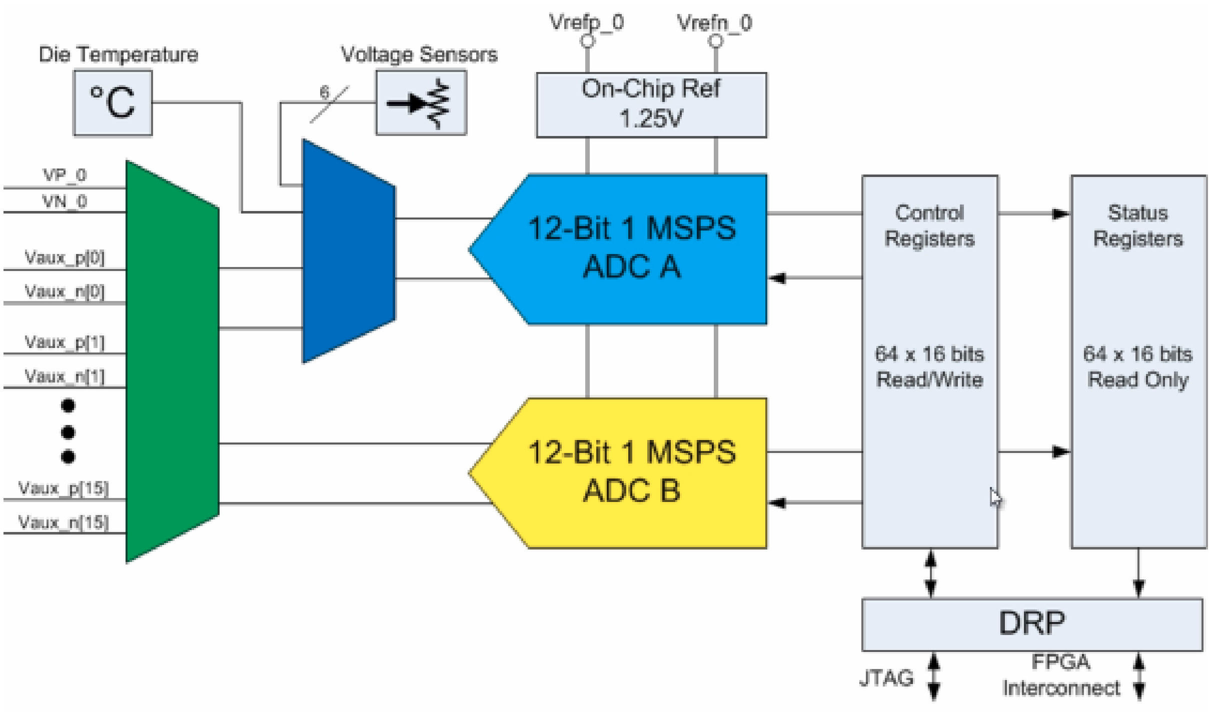


Figure 2.18: XADC Interfacing

The chosen FPGA board has an on-chip analog-to-digital converter (XADC), which is a **12-bit dual channel ADC** with a maximum sampling rate **1 Mega-samples per second (1MSPS)**. From Figure 2.18, we can see that the XADC supports upto 16 auxillary channels along with a dedicated analog channel with better sampling rate. The converted digital output of each channel is stored in status registers. The register address for Control and Status registers are mentioned in the Figure 2.19. For example, choosing auxillary channel 10 would result in storing the values in register with address **0x1A** . This project requires sampling and conversion of two analog channels, therefore XADC must be configured in ***Channel sequencer*** mode where each channel is sampled one after another in a loop.

However, the output from the module is a 16-bit data which just appends 4 bit zeros (in LSB) to the converted 12 bit value. As seen from Figure 2.20, the converted value its latched into status register only when **DRDY** signal goes high. Hence, when it is under multiple channel mode **DRDY** signal and register address specified in **CHANNEL**[**4:0**] signal must be used together to retrieve valid converted data values which is available on **DO**[**15:0**] wire. Xilinx's Vivado design suite which helps in programming this FPGA has wizards to configure the **XADC** settings and get the instantiation template to use

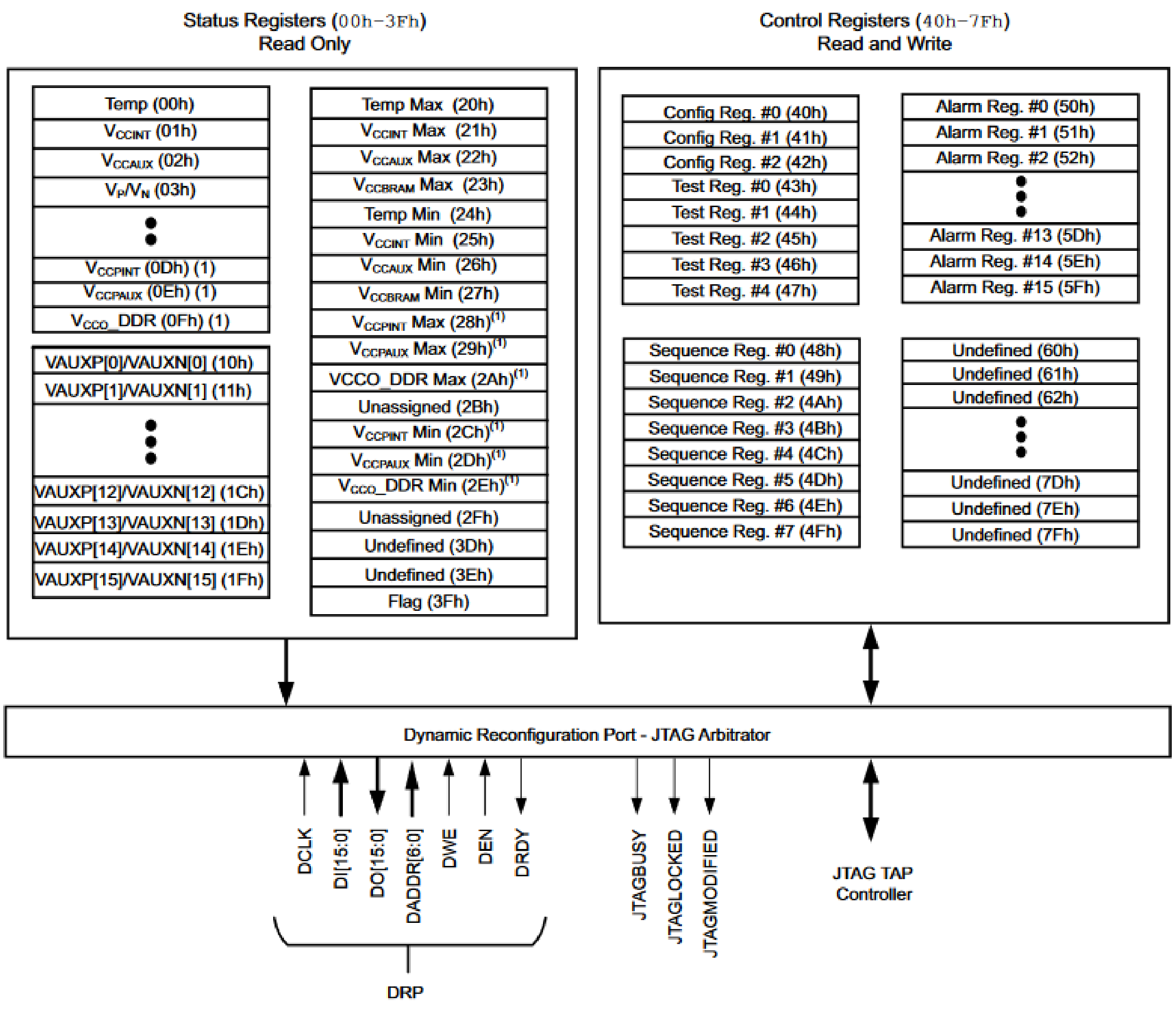


Figure 2.19: XADC Register Interface

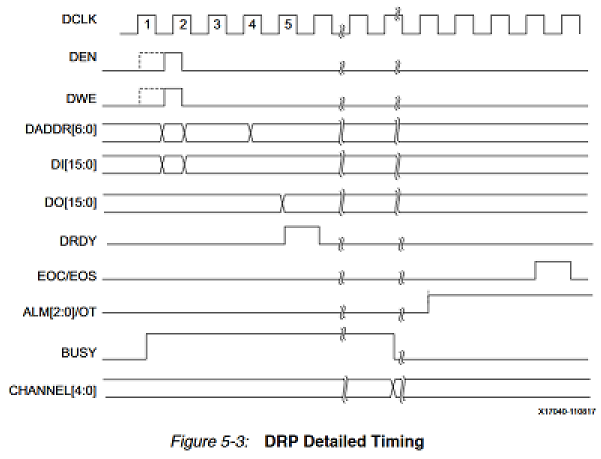


Figure 2.20: Detailed timing of events

with our design (code). ***Note:*** *XADC requires the input peak-peak voltage to be at max 1V, potentiometer can be used to tune the voltage levels accordingly*

2.4.2 Importing Design

The previously designed modules were imported here (Vivado design suite) with ease. Proper clock rates and bit-widths of input/output were checked carefully which are usually the common sources of error while implementing.

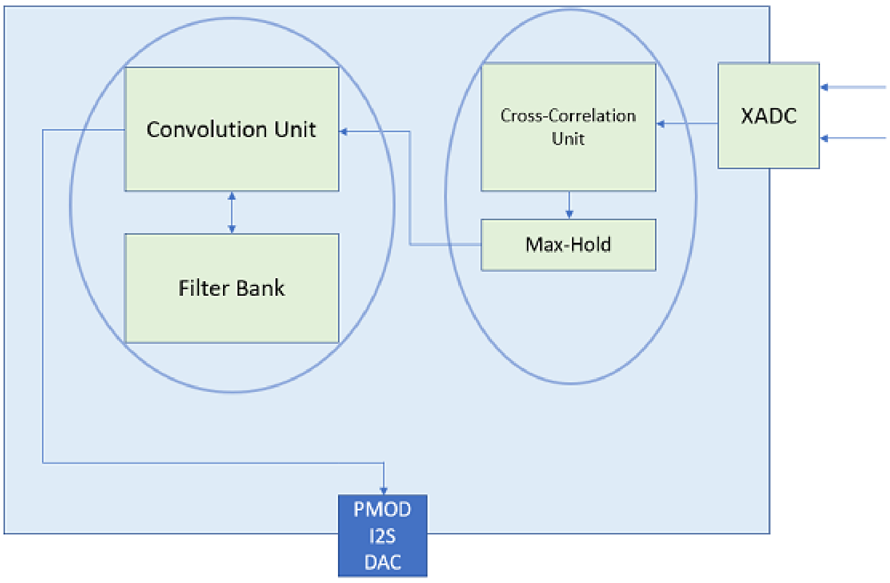


Figure 2.21: Block Representation

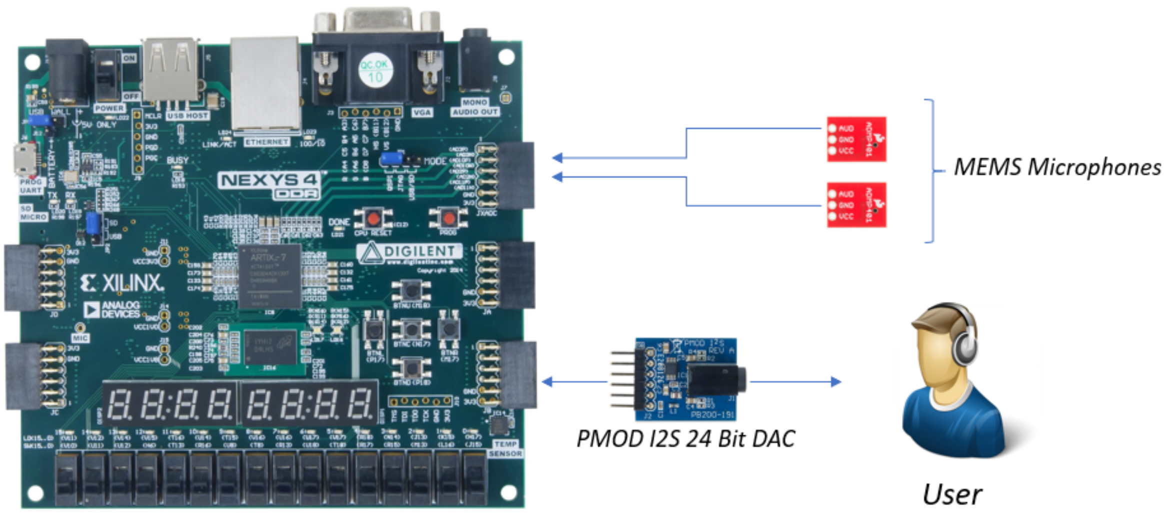


Figure 2.22: Setup

2.4.3 Output

The overall processed output of the entire design consists of 2-channel 16 bits data which needs to be converted to equivalent analog form and be heard through headphones. In order to implement this, DAC (Digital to Analog Conversion) unit is required. Hence, we have used an external **16-24 bit Stereo Pmod i2S DAC** module. It follows the *I2S* protocol which is typically used when dealing with audio signals. Thereby, we need to also introduce a block in our design code to include this transmission process.

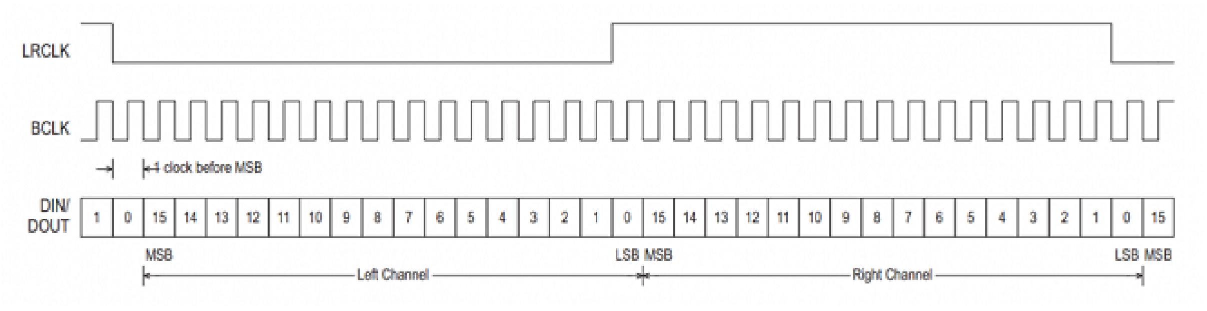
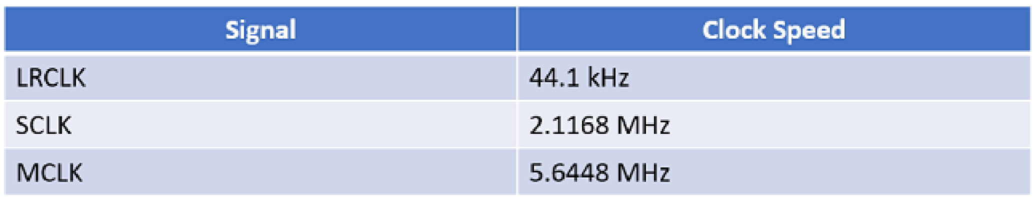


Figure 2.23: I2S Protocol



Signal	Clock Speed
LRCLK	44.1 kHz
SCLK	2.1168 MHz
MCLK	5.6448 MHz

Figure 2.24: Clock speed of different signals involved in I2S transmission

Xilinx's clocking wizard can be used effectively to get clock speeds ranging from 4 Mhz to 100 Mhz and later directly instantiate the same with ease. Here, in order to get clock speeds lesser than 4MHz we used clock dividers from the nearest multiple near 4MHz and derived it. From Figure 2.24, for example **_SCLK_: clock speed upto 4.2336 MHz** was generated from clock wizard and then halved manually. Typically, sending the data greater than sampling rate is advised since in that case you won't be missing any data to be transmitted. The PMOD module comes with a 3.5mm headphone jack which can be plugged in to listen to the output.

CHAPTER 3
Testing and Observations

Here are some of the observations, which were observed while experimenting with different audio samples and different HRTF filters.

3.1 Observation

3.1.1 Angle resolution

When the signals were correlated to get the difference in time of arrival for both microphones, for source angle range of 0° - 90° the number of distinct values (for Sampling rate = 44.1 kHz) was 26.

$$0^\circ - 90^\circ = 26(values) \tag{3.1}$$

3.1.2 Panning

Panning is the spread of a monaural signal in a stereo or multi-channel sound field. Here, we realize different directions of sound source by changing the filters accordingly. The resultant effect when heard will be equivalent to as if the source is moving around the head. This test was done in MATLAB to realise the functioning of HRTFs.

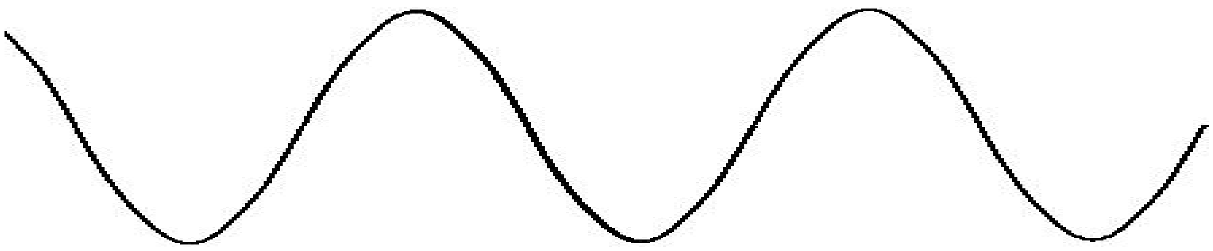


Figure 3.1: Input: Sine wave of 1kHz

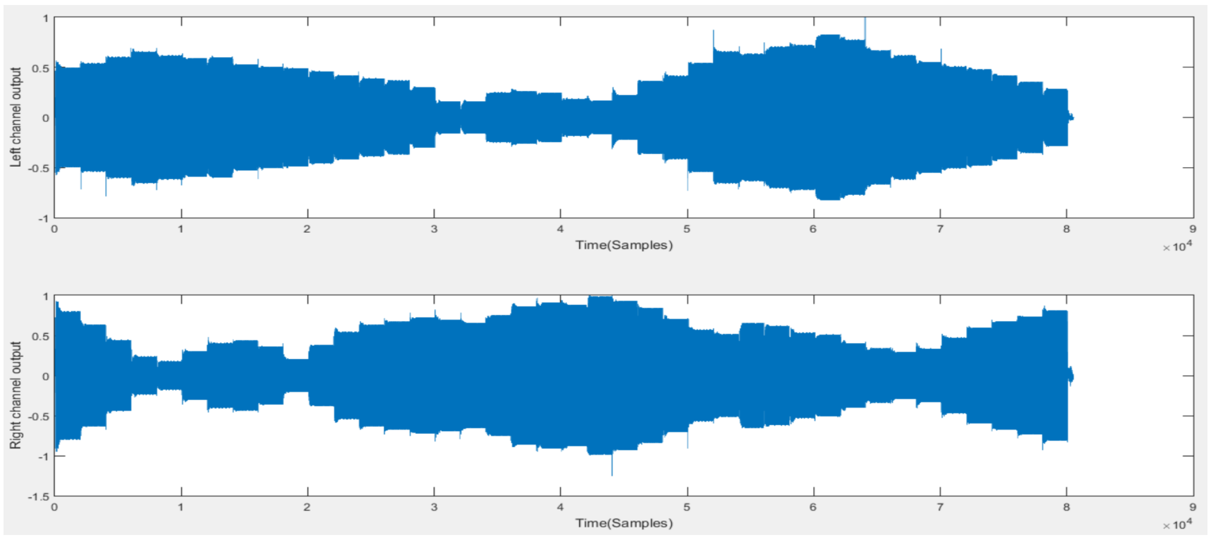
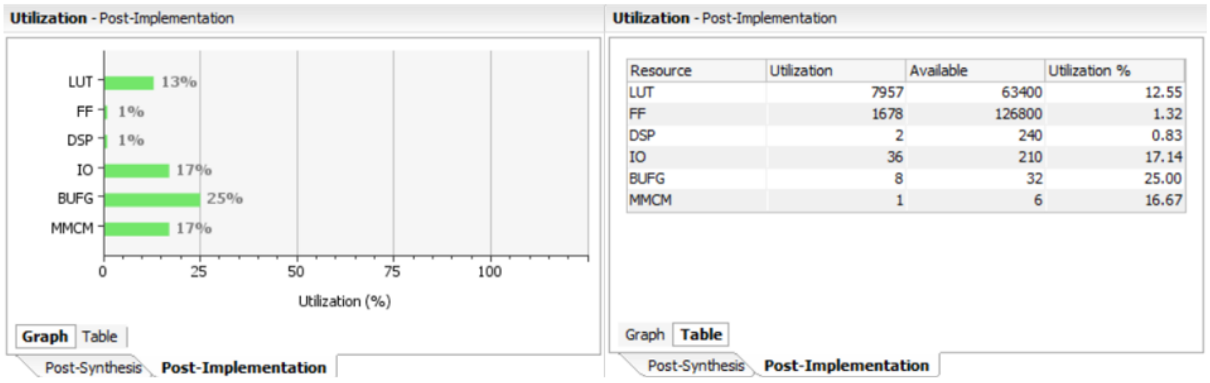


Figure 3.2: Output: After panning



Utilization - Post-Implementation

Resource	Utilization	Available	Utilization %
LUT	7957	63400	12.55
FF	1678	126800	1.32
DSP	2	240	0.83
IO	36	210	17.14
BUFG	8	32	25.00
MMCM	1	6	16.67

Figure 3.3: Resource Utilization

3.1.3 Testing Design

During the FPGA implementation we tried to generate the effect of **Panning** as a part of testing the filter output. This was implemented with, the user being able to select the angle of source and realise the effect. This test turned out to be successful with no problem in the output. Along with that , the **actual design** was also imported which detected proper **angle of sound source**. However, the design sometimes struggled under multiple sound sources. This can be said as one of the drawback of this design.

From the resource utilisation, Figure 3.3 we observed that most of the resources were taken up by storing the filter coefficients. If there was some kind of external memory interfacing used, the utilisation would have been significantly reduced. But as far as the quality and realisation of 3D sound is concerned the output was clear and as expected.

CHAPTER 4
Conclusions

4.1 HRTFs

After working with such filters we can easily say that HRTFs are a convenient way to realise the simulated effect of 3D surround sound. The change of filters differing in the azimuthal angle indeed causes a large difference in the audio output. Where as , change in elevation didn't cause much of impact. This can also be because of the *non-individualised* filter response. This means that HRTFs usually works the best when the filter responses are actually recorded with one's own facial structure. This is mainly because of the greater impact of parameters such as *ear shape, size, torso* matters in the overall hearing response. It was also observed that the filter length could be as low as **64**, below which resulted in compromise in audio quality. We always aimed to have the number of filter coefficients as **low** as possible to reduce the number of **MAC operations** but also wanted to have a **sharp** frequency response. A filter length of **100** proved to be optimal in its performance.

4.2 Analysis of MATLAB implementation

MATLAB being a handy tool helped in quick design checks and simulation. Although any kind of testing there would be post-processing but, helped in giving more insights about the process in terms of **plots, spectrum** etc.

In order to verify the output of the RTL design, scripts were written to mimic the exact process here which accelerated the process of testing. The only difference being that the MATLAB was working on **floating point** represented numbers where as in the hardware side all sorts of data were represented in **fixed point format**.

4.3 Analysis of RTL Design

As mentioned before during the course of this project there were 2 versions of RTL design, which was the resultant of testing them in real time and realising the shortcomings.

4.3.1 Analysis of Design - Version 1.0

The *Version 1.0* had the following design characteristics:

- Correlation of 30 samples
- Calculating delay in consecutive data frames
- 64 tap filters
- 13 different filters for particular elevation

The issues with this design as mentioned in section 2.3.3 had to be addressed and resolved. Although this design required less number of computations but, they had to be done in a quick interval of time and was idle for the most part of it. Hence not completely utilising the time available for calculation. This was one of the main reason giving rise to higher clock speed requirement.

4.3.2 Analysis of Design - Version 2.0

After learning that the rate of correlation and calculation of angle of sound source can be relaxed, changes were made and represented in *Version 2.0*. The design characteristics are :

- Correlation of 100 samples
- Calculating delay in every **5th** data frame
- Computing period of **4** data frames
- 100 tap filters
- 25 different filters for particular elevation

This design resulted in better control for selection of filters. One of the main aspects of this architecture was to carefully transit between filters producing smooth output. This was explained in detail in section 2.3.3. The overall output observed was as expected with **proper angle of sound source detection and change of filters accordingly**.

4.4 Final Inference

Given that the aim of this project was to enable the user to **realise 3D surround sound** only through a pair of earphones, the designed system serves the purpose and functions in **real time**. Based on subjective listening test it was verified that the user was indeed able to locate and the visualise the sound source **emerging from a particular direction**.

4.5 Future Work

There are certain drawbacks in this system which needs to be looked upon. The following can be the upgrades or modifications:

- Modify the system to include the presence of multiple sound sources
- Real time tracking of head movements to dynamically update the filter selection
- Emphasize the effect of elevation

REFERENCES

[1] B. Van Den Broeck, A. Bertrand, P. Karsmakers, B. Vanrumste, H. Van hamme and M. Moonen, "Time-domain generalized cross correlation phase transform sound source localization for small microphone arrays," 2012 5th European DSP Education and Research Conference (EDERC), Amsterdam, 2012, pp. 76-80.

[2] C. Knapp, G. Carter, "The generalized correlation method for estimation of time delay,"IEEE Transactions on Acoustics Speech and Signal Processing, vol. 24(4), pp. 320-327, 1976.

[3] B. Kwon, Y. Park and Y. Park, "Analysis of the GCC-PHAT technique for multiple sources," ICCAS 2010, Gyeonggi-do, 2010, pp. 2070-2073.

[4] E. M. Wenzel, Localization in virtual acoustic displays, Presence, vol. 1, pp. 80 107, Winter 1992.

[5] Brandtsegg, Φ.; Saue,S. Live Convolution with Time-Variant Impulse Response , DAFx-17, September 2017.

[6] Chong-Jin Tan and Woon-Seng Gan, User-defined spectral manipulation of HRTF for improved localization in 3D sound systems, IEEE electronic letters, vol. 34, pp. 2387-2389, Dec. 1998.

[7] C. Phillip Brown and Richard O. Duda, A Structural Model for Binaural Sound Synthesis, IEEE Transactions on speech and audio processing, VOL. 6, NO. 5, 1998

[8] Hans Wallach, "The role of head movements and vestibular and visual cues in sound localization", Journal of experimental psychology, Vol. 27 (4)