DAT096

Ultrasonic Sound Localization using Microphone Arrays

Group M3

Ashwin Kumar Balakrishnan
John Croft
Prema Manickavasagam
Hezhe Xiao

May, 2020

CHALMERS
UNIVERSITY OF TECHNOLOGY

Abstract

From echolocation used in SONAR to ant colony optimization techniques used in artificial intelligence, engineers have always tried to incorporate natural phenomena for innovative solutions. This project is one such example where the system designed detects ultrasonic sources in two dimensional space using linear arrays of MEMS microphones. It uses a signal-processing technique called beamforming which is implemented on an Artix-7 FPGA. The array in its current implementation consists of 4 microphones with a sampling frequency of 3.2 MHz and a source frequency of 25 kHz.

Signal-energy corresponding to particular directions is computed onboard the FPGA, but currently suffers from implementation-specific data discrepancies, which prevents further analysis.

Contents

Acknowledgements

We would like to thank Lena Peterson for her inexhaustable enthusiasm and support, as well as Sven Knutsson and Bhavishya Goel for lending their advice and technical expertise. Finally, we would also like to thank Syntronic AB for providing the sensor hardware used throughout this project.

Acronyms

CIC Cascaded Integrator Comb. 12

DSP Digital Signal Processing. 1, 7

EDA Electronic Design Automation. 7

FIFO First-In-First-Out. 17

FIR Finite Impulse Response. 12

FPGA Field-Programmable Gate Array. 7

HSMC High-Speed Mezzanine Card. 9

IP Internet Protocol. 17

LUT Look-up table. 7

PC Personal Computer. 10

PCM Pulse-Coded Modulation. 10

PDM Pulse-Density Modulation. 10

RAM Random Access Memory. 15

ROM Read only Memory. 13

CHAPTER 1 Introduction

The main aim of this project is to locate an ultrasonic acoustic source in two dimensional space using a linear array of MEMS microphone sensors. This is done using a technique called beamforming, in order to achieve directionality in signal reception.

The project was executed according to SCRUM principles, in order to deliver a product of the highest possible value. The entire duration of the project was divided into 5 *sprints*, each with a duration of approximately three weeks, where goals were set at the beginning of each sprint and at the end of each sprint, a *Sprint Restrospective* was conducted, where the success and shortcomings of the sprint were analyzed in order to improve efficiency for the subsequent sprint. This iterative methodology is intended to quickly force teams to develop. Version control for all software developed in the project was done with `git`.

The most common application that comes to mind when discussing ultrasonic source localization is how bats use ultrasonic sound waves [1] to locate objects in space. This concept was adapted in SONAR [2], used by submarines to navigate underwater and communicate by determining the direction and distance of the transmitter. The problem in this project is similar, but does not use echolocation or radar (it is strictly source to receiver) and does not detect distance.

1.1 Problem description

The task in this project is to design an FPGA-based Digital Signal Processing (DSP) system that tracks the location of a mobile acoustic source in two dimensions using a technique known as *beamforming*. The final application of the system is to track robots equipped with such an acoustic source. A beamformer is, in essence, a spatial filter that boosts the signal from the *look-direction* while attenuating signals from all other directions. The system to be designed should operate in real time and the direction of the beamformer should dynamically change to locate the source.

1.2 System Requirement Specifications

The system to be designed should not contain any movable parts and should be able to :

- track the source in two dimenstional space.
- detect change in position of the source with a latency of less than one second.
- detect the source movements of more than two degrees within the cone of interest.
- display the real time co-ordinates of the source on a PC.

The system may assume a fixed-frequency audio source.

1.3 Scope

In this project, the goal is to implement beamforming using the delay-and-sum technique first and foremost. If that is successful and there is a reasonable amount of time left before the final evaluation, other techiniques may be taken into consideration.

CHAPTER 2 Background

2.1 Beamforming

Beamforming is a type of spatial filtering in which a signal receiver is more sensitive to signals from certain directions, while attenuating signals from other directions. It uses an array of smaller omnidirectional sensors to, in essence, emulate a larger, directional sensor.

This is in contrast to an omnidirectional receiver which is equally sensitive to signals from all directions, and hence performs no spatial filtering whatsoever. The basic principle of operation is that the receiver is configured in some way (in terms of DSP operations and physical geometry of the sensor) to constructively interfere signals from the desired direction while destructively interfering signals from all others.

By configuring the DSP part of the receiver, the direction of sensitivity can be steered without physically moving the sensor array. In one dimension, the sensitivity can conveniently be represented in a polar plot showing gain versus incident angle ψ. This is referred to as a *beam pattern* and an example can be seen in fig. 2.1.

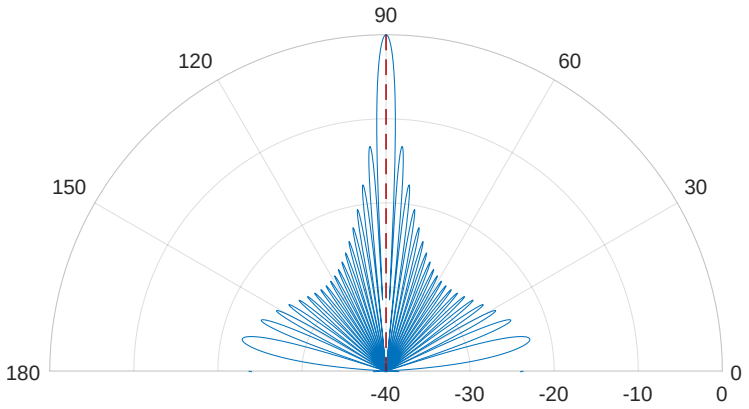

Figure 2.1: Beampattern illustration showing mainlobe, sidelobes for a source frequency of 25kHz and 24 microphones

The beam in the direction of sensitivity is referred to as the mainlobe. There are usually prominent, but attenuated, side-lobes on either side and, in the case of aliasing, which is discussed below, so-called grating-lobes which are unattenuated.

The main lobe is characterised by its *beamwidth*, which is the range of values for which it is largely unattenuated, and which will affect the accuracy of a system designed to locate a signal source.

2.2 Beamforming Techniques

There are two types of beamforming technique: fixed and adaptive beamforming. Fixed techniques are based on the fixed elements whereas adaptive techniques include a feedback mechanism which can adapt to the surrounding environment. Fixed beamformers are easier to implement in comparison. Some adaptive beamforming algorithms are Least Mean Square (LMS), Normalized Least Mean Square (NLMS), Sample Matrix Inversion (SMI), Recursive Least Square (RLS) and Hybrid Least Mean Square/Sample Matrix Inversion (LMS/SMI). In these algorithms, the signals are assigned *weights* and they are updated based on the *error* signal which is the difference between the desired value and actual signal value [3]. In a fixed beamformer, a main lobe in the desired direction is created and attenuates undesired signals from other angles. The filters are tuned to a particular value and do not adapt to external information. The delay-and-sum (DAS) technique is a fixed beamforming technique and was chosen for this project due to its relatively low complexity.

2.3 Delay-and-Sum (DAS)

DAS is a technique for steering the directional sensitivity of a sensor array. In terms of waves, an incident wavefront will reach each individual sensor in a linear array at a different time (provided the signal does not originate perpendicular to the array axis, in which case all sensors receive the wavefront at once). In order to steer the directional sensitivity, the signal that each sensor receives is delayed by an amount of time corresponding to the desired direction. By doing this, the physical delay between sensors for a given incident angle can be compensated for. The delayed signal samples are then summed to give either constructive or destructive interference. If the delays match the incident angle (i.e. the sensor array is configured to "look" in that direction), each sensor will delay its received signal such that they are all in phase and maximum value is obtained when they are constructively summed.

Weights may also be used in conjunction with delay shifts, affecting the beam pattern, though that is out-of-scope for this project.

2.3.1 Sensor Arrays

Sensor arrays are receivers with multiple sensors arranged in a specific way. The most basic type of sensor array is a equidistant linear array, in which sensor elements are positioned along an axis with equidistant spacing between them.

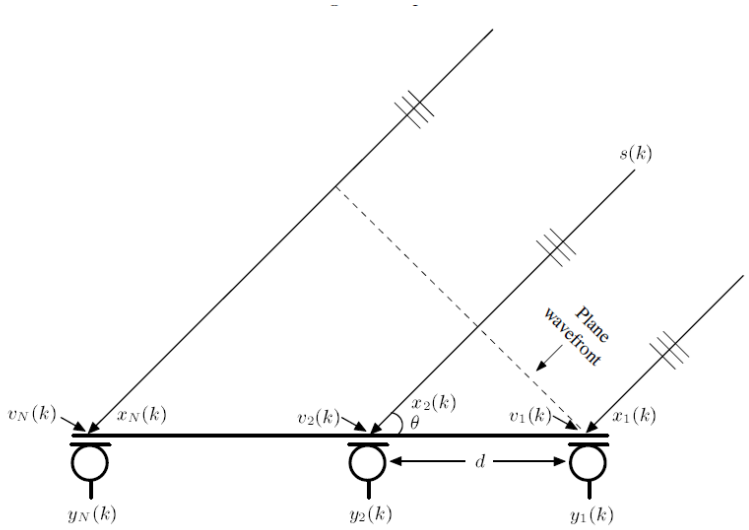

Figure 2.2: In a linear array of sensors, a plane wavefront will reach each adjacent sensor with a time, and correspondingly, phase offset. Source in [4].

In this type of sensor array, an incident wavefront will reach each sensor element at a different time, resulting in a phase shift on each. This is illustrated in fig. 2.2. A linear array can only direct its sensitivity in one dimension.

By combining multiple linear arrays, placed orthogonally to each other, it is possible to extend directionality to two or three spatial dimensions.

Other types of sensor array also exist, in particular composite arrays which are a simple way of partitioning a source signal containing multiple frequencies (broadband) into sub-bands that can be filtered individually [5].

Likewise, several algorithms exist for the DSP part of the beamformer. A simple, but naive, approach is delay-and-sum (DAS). This approach is the one adopted for this project. DAS is naive in the sense that it does not use the values it produces.

More adaptive approaches, for example using a least-squares error minimization technique, use the values produced by the receiver as feedback to try to optimize some particular variable, such as signal strength, by steering the directional sensitivity. Ostensibly superior in terms of system performance, such algorithms come at a cost of significantly more difficult theoretical and implementation challenges.

2.3.2 Spatial Aliasing

As previously mentioned, the spatial separation of sensor elements in a particular sensor array results in a relative phase shift in the received wavefront in each sensor. When the incident angle and spatial separation are such that the phase shift is one wavelength λ or a multiple thereof, the direction becomes ambiguous. This is shown in fig. 2.3.

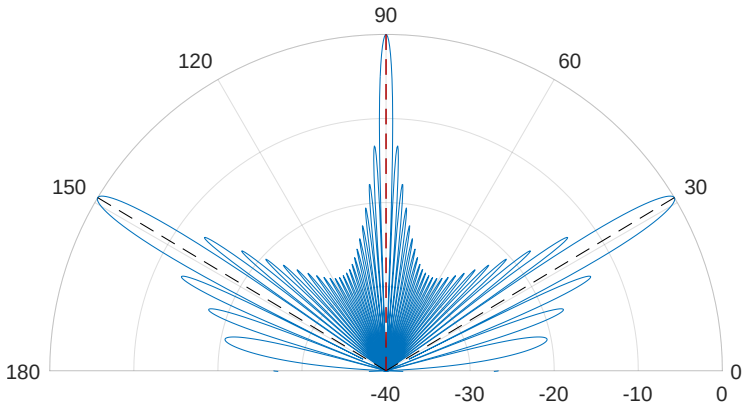

Figure 2.3: Incident angles that result in a phase shift of a wavelength multiple result in spatial aliasing. These appear as grating lobes in the beampattern.

Aliasing is obvious in a beam pattern as the lobes are unattenuated.

2.4 Audio Location

In order to know from which direction a signal is strongest (and thus, in all likelihood, determine the direction of the source), DAS beamforming is not enough. The summed output signal will be a sinusoid with a magnitude corresponding the beampattern for a given configuration. To be able to compare these magnitudes, the signal *energy* must be calculated.

The energy of a signal is calculated using the following formula:

$$E_c = \int_{-\infty}^{\infty} \mid x_c(t) \mid^2 \; dt \tag{2.1}$$

where $x_c(t)$ is a continuous signal. Equivalently, for a discrete signal

$$E_d = \sum_{-\infty}^{\infty} |x[n]|^2 \tag{2.2}$$

This could be calculated for every angle, with the one which has the highest energy value being the direction of arrival. After a full sweep across the entire range of values from two microphone arrays placed perpendicular to each other are multiplied to get the highest energy signal and locate the source.[6]

CHAPTER 3

Hardware Design

3.1 Xilinx FPGA Artix-7 AC701

Field-Programmable Gate Array (FPGA) is a semiconductor integrated circuit consisting of configurable logic blocks and interconnects [7]. The logic blocks consist of components like flip-flops and lookup-tables (LUTs) as well as specialised Digital Signal Processing (DSP) circuits such as multiply-accumulators. The application range for FPGA is very wide because of its flexibility. They are programmed by hardware description languages to achieve the desired function.

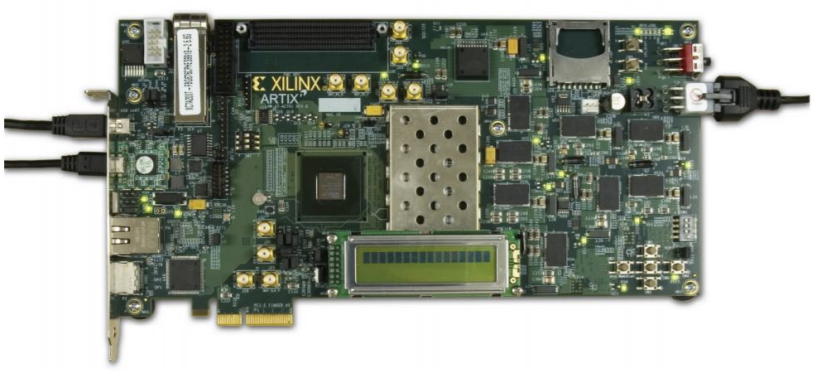

Figure 3.1: FPGA Artix-7 AC701

FPGAs are inherently parallel and hence suited to carrying out various functions independant of each other without having to contend for resources (such as a single processor) [8]. The hardware platform used in this project is the Xilinx AC701 Evaluation Kit (shown in fig. 3.1), using an XC7A200T FPGA and various peripheral devices such as DDR3 RAM. Xilinx Vivado is the main Electronic Design Automation (EDA) tool used for design and synthesis.

3.1.1 FPGA: inherent advantages and disadvantages

The inherent parallel computing of FPGA's make it faster and more efficient than other serial processors where the computation speed is low when compared

to the former. It is flexible because of the logic blocks which can be programmed to carry out the desired function. FPGA's can be reprogrammed easily without the need to rewire and the functionality can be changed faster. This feature also enables downloading algorithms and change them just like a computer changes program [9]. They are reusable and hence is very cost efficient and makes them ideal for prototyping. It eases up the product development and takes less time to market. They can be implemented in various real time systems because of its efficient architecture and high computation speed [10].

FPGAs have a comparatively expensive unit price and designing for them is inarguably more cumbersome than a purely software approach. Some FPGA devices include processor cores on-chip, enabling a traditional software envionment (such as a Linux based operating system) to interface with the reconfigurable FPGA fabric. This approach tries to leverage the ease of programmability of an established hard-core processor (such as an ARM core) along with the extreme design flexibility of the FPGA.

The FPGA used in this project does not have a built-in processor, though there exists Xilinx IP modules (Microblaze) to instantiate "soft" cores, processors that are constructed from the FPGA fabric directly. These are generally slower and consume area depending on their configuration, but otherwise fulfil the same purpose.

Furthermore, FPGAs have many IO connections (the XC7A200T-2FBG676C FPGA onboard the AC701 Evaluation Kit has 400 such pins), making them ideal for applications requiring many peripheral sensors and devices.

3.2 Microphone Array

The microphone array used in this project, shown in fig. 3.2, is designed by *Syntronic* and it consists of a linear array of 24 MEMS microphones.The distance between the microphones play a vital role in determining the frequency it can be optimized for. In the array used in this project the microphones are

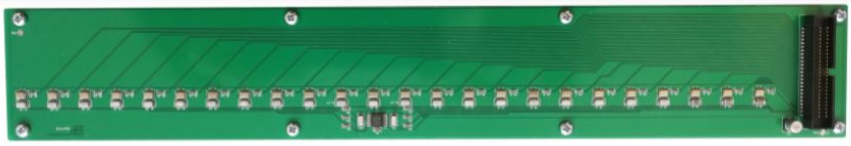

Figure 3.2: Microphone array

spaced 11.43mm apart. The microphone used is SPH0641LU4H-1, which can be clocked at different frequencies to get different bandwidth. It is a miniature high performance microphone with one bit PDM output. This has to be filtered inorder to get a PCM output. It has RF immunity, has a high SNR of 64.3dB and supports ultrasonic applications [11].

3.3 Breakout board

The breakout board, shown in fig. 3.3, designed by *Syntronic*, acts as a bridge between the FPGA and the sensor-array board. It uses the High-Speed Mezzanine Card (HSMC) connector on the AC701 to route data signals and a single clock signal to 4 sensor-array breakout connectors.

The breakout board includes a `PI6C49X0206T` fanout-buffer that drives the clock signal to all four sensor arrays.

Since each sensor-array has 24 sensors, the FPGA must potentially drive 97 pins (accounting for the single clock signal), for which the density of the HSMC connector is particularly suited.

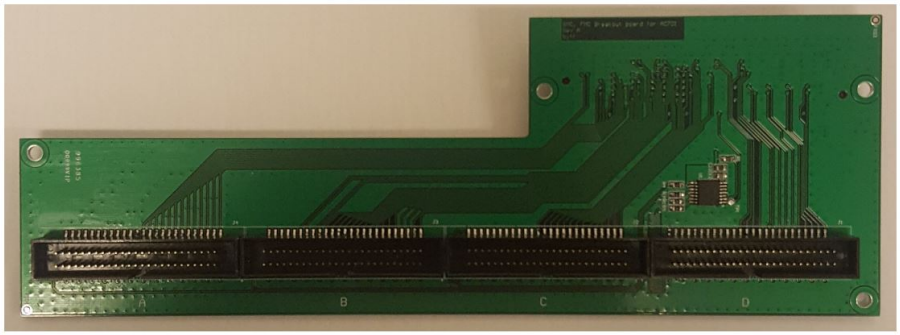

Figure 3.3: FMC Breakout board

3.4 Ultrasonic Audio Source

The audio source used in this project is a commercial "dog-whistle", a handheld tone-generator with a directional beam that produces a relatively pure 25 kHz frequency. This tone, though great in volume, is imperceptible to humans but within the audible range for dogs, hence the name.

CHAPTER 4 Implementation

A reference HDL design was provided, which implements a number of essential hardware structures, laying the basis for the rest of the project. Most notably, a modified version of the Cobham-Gaisler Leon3 processor is implemented, along with peripheral devices such as various controllers for busses, RAM and ethernet as well as a debug interface that is accessable using the debug monitor GRMON, also from Cobham-Gaisler. All hardware IPs are provided via the GRLIB IP library.

Additionally, the reference design implements a filter-cascade that converts raw Pulse-Density Modulation (PDM) data from the microphones to 16-bit PCM data , as well as a FIFO to buffer data until it can be written to onboard RAM.

The intended data flow of the reference design is that data is sampled from the microphones, processed in some way that implements beamforming and then stored sequentially in RAM. This stored data is then accessed from a conventional PC via ethernet using the GRMON debug monitor, which directly accesses RAM contents.

4.1 System Architecture

The system is built on the already given reference design responsible for simultaneously taking multiple input from four microphones and implementing acoustic source localization.

The multiple signals are transmitted through one channel via a multiplexer (mux), in 1-bit Pulse Density Modulation (PDM) format, to be converted into a 16-bit Pulse-Code Modulation (PCM) signal before it can be processed as shown in fig. 4.1 below. After conversion, the interleaved data is separated using a demulitplexer (demux) and a bitstream containing the number of clock cycles corresponding to the delay values for each scanning angle is driven to the input address of the shift register. These delay values of microphones for each angle are pre-calculated and saved in Read-Only Memory (ROM). After summing the delayed waveforms, the pressure-energy of the signal is calculated, finding the peak value (in order to obtain the total energy) and sending it to the PC for further analysis to get the acoustic source direction.

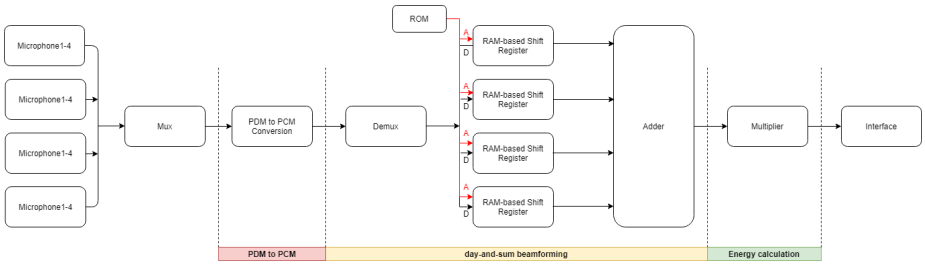

Figure 4.1: System architecture diagram. For details, see text.

4.2 PDM to PCM conversion

The given reference design implements the conversion from PDM to PCM using one cascaded integrator-comb decimating filter(CIC filter), one half-band FIR filter, one low-pass FIR filter and one high-pass filter to output the 16-bit filtered data at a 100 kHz rate.

As the fig. 4.2 shown, CIC Filter IP block is used to decimate the PDM input by a factor of 16 and to convert it into a PCM format, Halfband FIR Filter IP block decreases the sample rate of CIC output by 2 and compensates for the passsband of CIC filter which is not flat [6]. Low-pass FIR filter IP block passes the low frequency signal components and removes the high frequency noise and high-pass filter is the final stage to remove any DC component induced by the sigma-delta modulator or the microphone itself (offset error).

Since the input sampling rate of filters depends on the input clock of the microphone system, the design has to generate appropriate clock inside the FPGA. The sampling frequency of PDM is 3.2 MHz(100 Khz*32) based on the PCM sampling rate of 100 KHz of this design but because of the hardware limitations, it cannot generate 3.2 MHz clock directly from 100 MHz as the input clock from Microphone system. Therefore, clock dividers are used to generate a clock frequency of 3.2 MHz. Now this design is extended to 4 microphones by using the multichannel mode of the CIC and FIR filters.

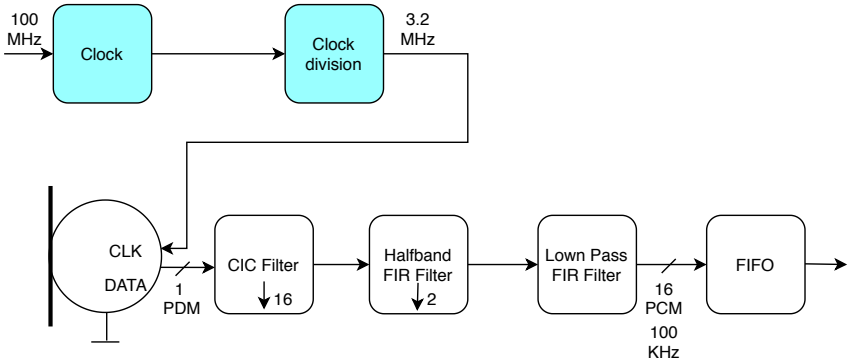

Figure 4.2: Design clocking

4.3 Multi-channel implementation

4.3.1 Filters

The reference design is extended for 4 microphones by using the multiple channel mode of the CIC and FIR filters, so that the muxed signal can be processed in a serial manner.

4.3.2 Accesing multiple Channels of the CIC and FIR filters

The muxed PDM data from multiple microphones is sent as input to the CIC filter Data Input Channel. From CIC filter schematic (fig 4.3), there are two channels, the channel master and the channel slave, to control CIC Compiler core's processing of data. TVALID is driven by the channel master to show that it has data to transfer and TREADY is given by the channel slave to show that it is ready to accept data. The transfer happens until both TVALID and TREADY are high [12].

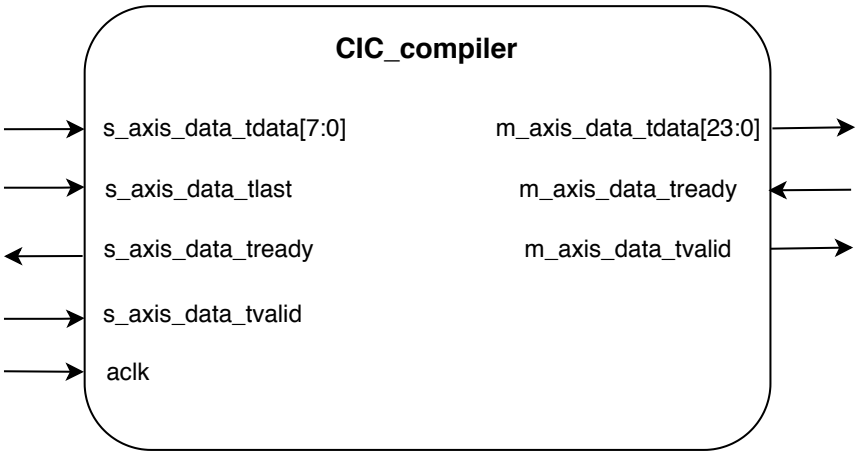

Figure 4.3: CIC filter schematic symbol

TDATA is the Data Input Channel and carries the unprocessed sample data. TLAST is also for the Data Input Channel and it is only used in multichannel mode, asserted by the sample data corresponding to the last channel. The TLAST of the input channel is set to 1 after data from all the channels is sent. All of these use the AXI4-Stream protocol [12]. Fig. 4.4 shows this protocol of CIC filter input with four channels. The design of multiple channels IP block mainly depends on it. The same technique is used for FIR filters also.

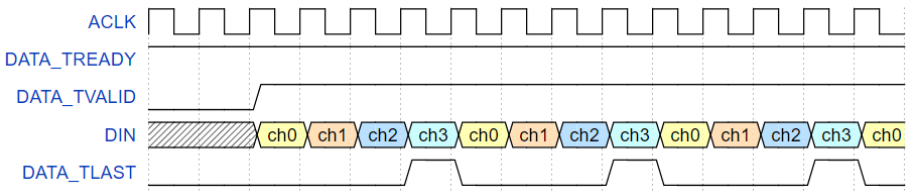

Figure 4.4: Timing diagram of CIC filter input

4.3.3 De-Multiplexer

After the PDM data is successfully converted to PCM, we separate the data from their muxed form by using a demux. The interleaved data is seperated at the rising edge of the clock cycle again and stored in registers for DAS beamforming.

4.4 DAS beamforming

The delay values for each microphone with angles from 30 degrees to 150 degrees are pre-calculated in MATLAB. Assuming the source is in the far field and sound speed $\boldsymbol{c}$ is 343 m/s, $\boldsymbol{d}$ is the space distance between two neighboring sensors. The souce wave arrives one microphone with an angle of $\boldsymbol{\theta}$. The delay values between the $\boldsymbol{n}$:th microphone and the reference sensors is decribed by the flowing fomula [4].

$$\text{Delay} = \frac{(N-1)\,d}{c}\cos\theta$$

Using the calculation of $N_{clk} = ceil\,(Delay * f_{clk})$, f_{clk} is the sampling frequency of PDM of 3.2 MHz, to convert these daly values to a number of clock cycles and saving them in a COE file as the initialization of ROMs IP block in Vivado.

In the next experiment step, the clock frequency will be divided to drive the shift register IP blocks and using an adder IP block to add the maximum delay with the shift registers traversed by enable signals of FIR filters.

The formula below is used to obtain the plot of beam pattern with 24 microphones and sound frequency. $\boldsymbol{W}$ is the array's reponse to a signal, $\boldsymbol{f}$ is the sound frequency and $\boldsymbol{\Psi}$ is a directional angle from 30 to 150 degree (the incident angle of the wavefront) and $\boldsymbol{\theta}$ is the *look-direction* of the beamformer[4].

$$|W(\psi,\theta)| = \left|\frac{\sin\left(N\pi\frac{fd(\cos\psi-\cos\theta)}{c}\right)}{N\sin\left(\pi\frac{\mathrm{fd}(\cos\psi-\cos\theta)}{c}\right)}\right| \tag{4.1}$$

The array beam pattern can be seen in MATLAB with different numbers of microphones and value of source frequency, under the premise of fixed distance between two neighboring sensors, in order to help us analyse how the beamformer infected by the spatial aliasing. As we known, if the sidelobes are as low as possible, the signals from other directions except the source direction would be attenuated as much as possible so that we can determine the range of source

sound frequency, it is about from 27 kHz to 30 kHz by using dog whistle to avoid spatial aliasing.

In our system disgn we used four micros and dog wistle to test the sound location, the theoretical value of beamforming is showed below Fig. 4.5.

This figure plots the beam pattern for 4 microphones with 90° and f = 30 kHz. The mainlobe is from the sound direction properly at 90°.

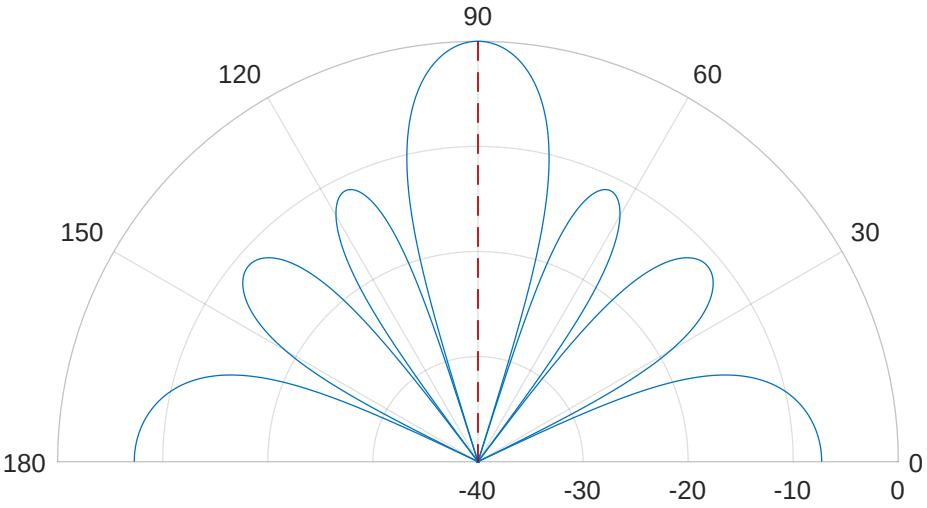

Figure 4.5: Beamforming of 4 microphones with frequency of 25 kHz at 90°

However, changing the direction of sound location, the beam pattern will adjust the angle of the mainlobe but also introduce new aliasing issues into the beamforming.

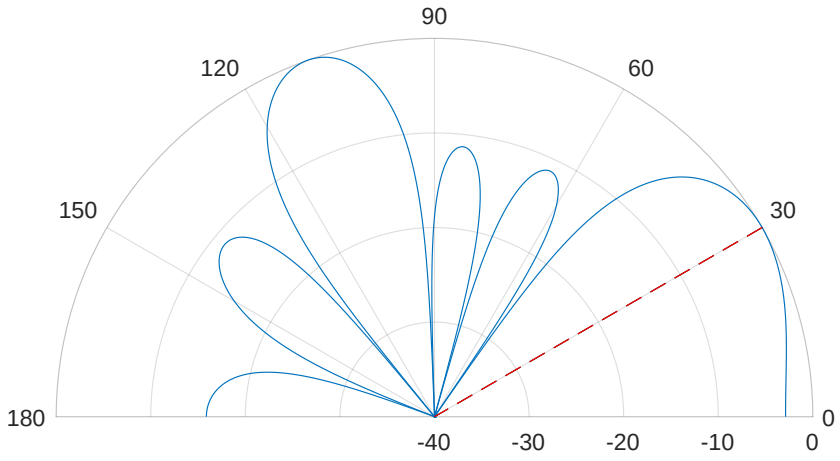

Figure 4.6: Beamforming of 4 microphones with frequency of 25 kHz at 30°

4.5 RAM based shift register and Block memory generator

The RAM based shift registers are used to delay the input signals by the values pre-calculated in matlab using the formula for finding the delays. It has an active high clock enable and the data from the demux is sent as the parallel data input. The delay values are stored in a single port ROM using a block memory generator in a .coe file. The values are accessed by the memory address and then outputted to the shift register. The delayed signals are then sent to the adder.

4.6 Finite state machine of scanning block

In the previous work, we already calculated the delay values of four microphones for the angels from 30° to 150° and we get the number of clock cycle to delay in shift registers for 120 angles, the maximum value of them is 278 (we use 300 as the mux cycle in the code of scanning block), which are saved in ROMs. Therefore, for the purpose of getting the directional energy of sound source, we have to adjust the delay at each angle, sample some data, add them together, square them to get the energy for directions and compare them to get the maximum value, which will be the direction of arrival. In order to control the shift register and energy calculation block, we need to use finite state machine, the design is shown below.

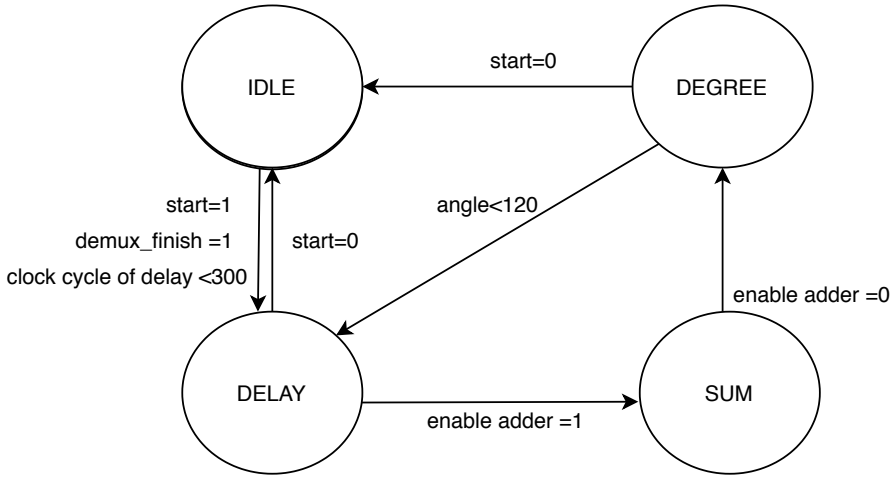

Figure 4.7: The state diagram for FSM to control DAS and compare the energy

The IDLE state is to wait for the switch on the FPGA turns on and all the data is finished to demux. The delay state is used for all the four microphones propagated out of the shift registers at each degree and the sum state is to add the amplitude together. Every time the system changes to the next four address

of next angle, the number of delay clock cycle is already saved in ROMs, it will return to delay value to assert a counter and read the new delay from ROMs.

In our system design, there are actullay 120 sets of number of delay clock cycle corresponding to a scan between 30° to 150°. Thus, the counter needs to increment 119 times to finish the entire scan for the angles in a system. We use a flag signal of scan_finish to indicate that all dalay values are implemented for four microphones and another flag signal of adder_enable to control the energy calculation blcok. The figure showed below illustrates how the micros are propagated out of the shift registers and energy calculation block is controlled by scaning block.

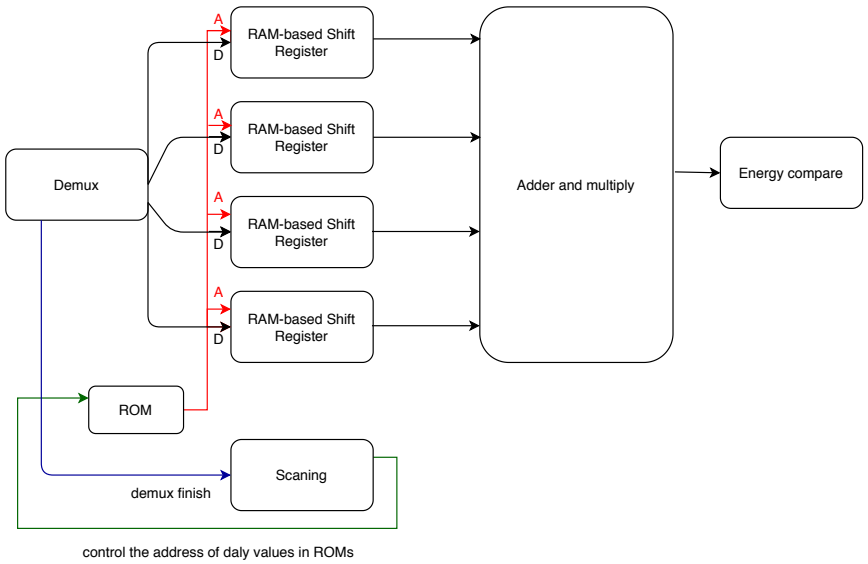

Figure 4.8: Diagram of how the micros are propagated out of the shift registers and energy calculation blcok is controlled by scaning blcok

4.7 Energy calculation and source location

The code for energy calculation is written in vhdl with sum and multiplication together and initialised as a block in vivado, which squares the input signal and the squared signals are compared to get the mux energy in order to identify the location of the sound source. The signal with highest energy (amplitude squared) is the desired ultrasonic source. The formula to calculate energy is showed in chapter 1 of background theory and prior work.

4.8 Data storage and communication (to PC)

As the system archtecture shown above, the PCM samples are saved in DDR on the board and read them from PC by Ethernet.

Therefore the memory controller, Ethernet controller and AHB bus are used to connect to FIFO IP block.

4.9 GRLIB and APB BUS

GRLIB is Gaisler Research Library to use for set reusable IP cores fro system-on-chip development using Gaisler's LEON3 processor, which is provided in the reference design.

It includes VHDL code for processor core, peripherals, bus and memory controllers and debug support unit to simulation and synthesis. LEON3 system is shown as following fig 4.9.

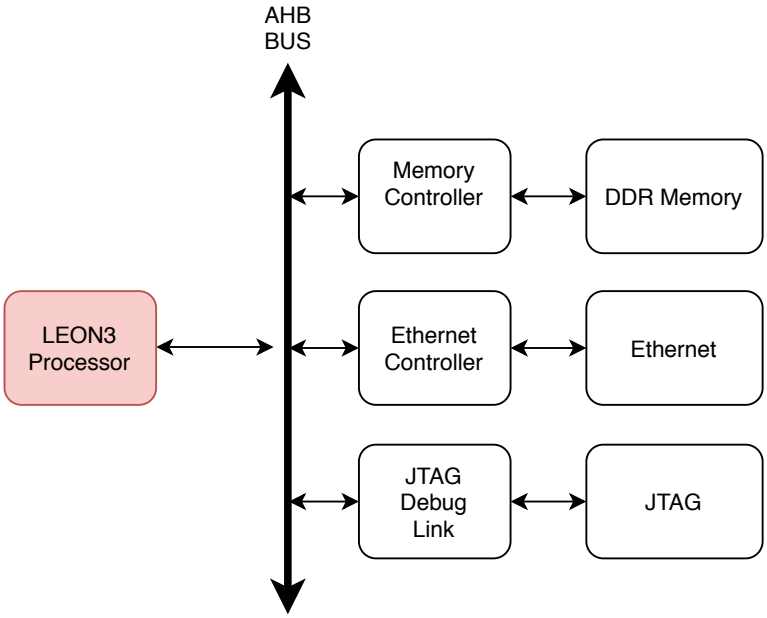

Figure 4.9: LEON3 based system

AHB Bus is AMBA High-performance Bus, which is multi-master bus for connecting peripherals with high data rates and variable latency. The structure is shown as fig 4.10.

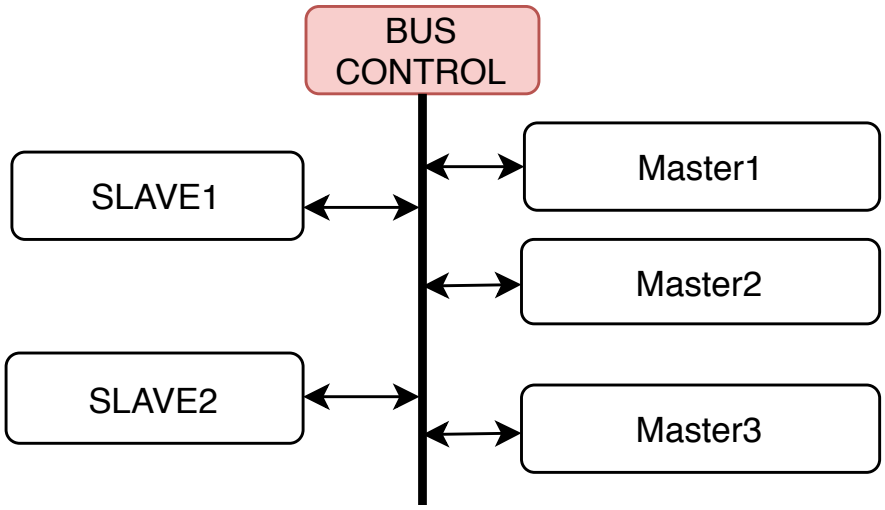

Figure 4.10: The structure of AHB Bus

CHAPTER 5 Experiments

5.1 Implementing multiple data channels and verifying in *Audacity*

All the microphones on the sensor array are clocked synchronously and thus output data in parallel. Multiple data channels were implemented to support this using time-multiplexing/demultiplexing (also known as interleaving/deinterleaving). The filter-cascade is composed of IP modules that can be configured to receive time-multiplexed data, and so only one filter-cascade needs to be instantiated.

Correct functionality was verified by recording a known frequency, importing the sampled data into *Audacity* (which also interprets multiple channels as time-multiplexed) and using the built-in spectrum analyzer to verify that the known frequency is reproduced in each channel.

5.2 Implemented support for ultrasonic mode in sensors with verification using spectrum analyzer in *Audacity*

To implement ultrasonic support, the data rate of the PCM signal on the output of the PDM-to-PCM filter-cascade must be increased to at least twice that of the ultrasonic audio source. Furthermore, the microphones require a minimum clock-rate of 3.072 MHz in order to sample ultrasonic frequencies. The PDM clock was adjusted to 3.2 MHz, which after decimation gives a PCM data-rate of 100 kHz, which is adequate to sample the 25 kHz audio source.

This necessitated computing new filter coefficients for the filter cascade, using the new sample rate with the existing decimation rates.

The correct functionality was verified by importing sampled data (without any processing) into *Audacity* and using the built-in spectrum analyzer. The result is shown in fig. 5.1, which shows a relatively pure tone with its peak at 24.985 kHz.

The figure of system design is showed as fig 5.2

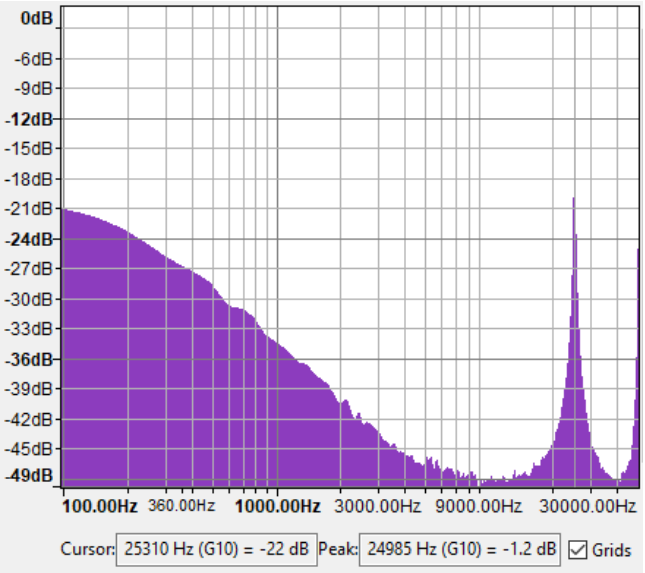

Figure 5.1: *Frequency spectrum of sampled ultrasonic signal source. Peak at* 24.985 kHz.

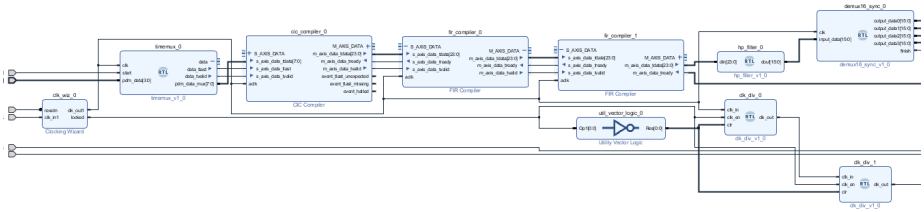

Figure 5.2: The partial design of mux and demux in vivado

5.3 Summation of multiple sensor inputs without delays (equivalent to look-direction of 90°)

The demultiplxed signals for four microphones are added together using adder IP blocks to test the look-direction of 90°. The sampling frequency of PDM is 1.4112 Mhz and the sampling frequency of PCM is 44100 hz according to the reference design first, which is verified to ultrasonic frequency of 3.2 Mhz next.The figures of system design and recorded raw data(human voice)in AUDACITY is showed as following.

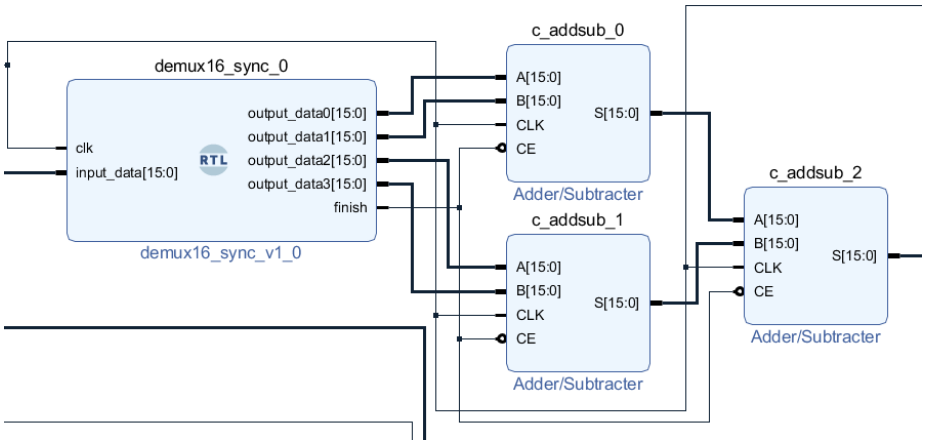

Figure 5.3: The partial design of demux and adders in vivado

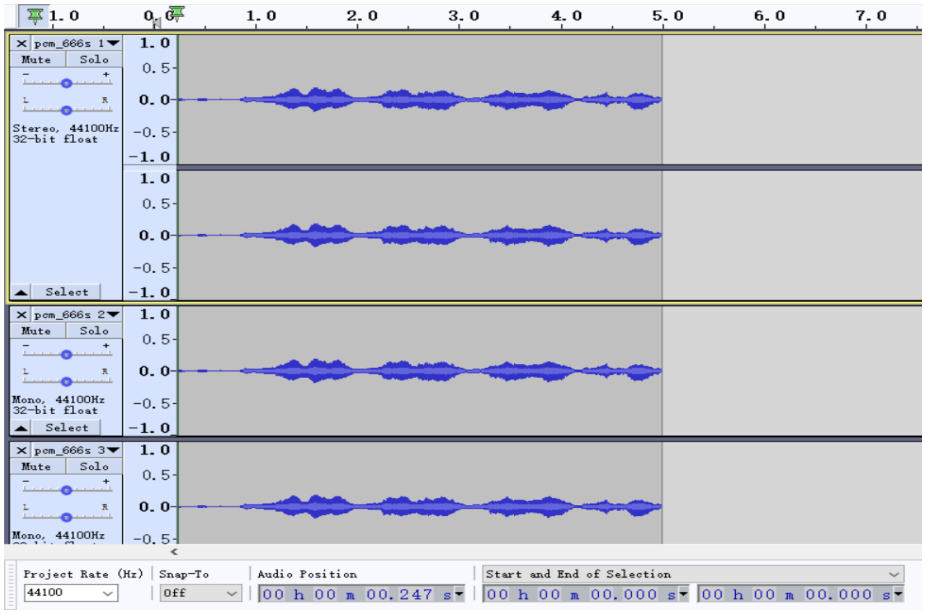

Figure 5.4: Raw data of 44100 hz of PCM implemented by demux and adder IP block in vivado

5.4 Summation of multiple sensor inputs with specific delays (equivalent to look-direction with several specfiec different angles)

After demuxltipling one signal to four 16-bit signals, adding four shift register blocks, four ROMs and three adders in the system design to point to the specific address in the ROMs of each angle. Firstly, seven angles were chosen, which are 30°, 50°, 70°, 90°, 110°, 130°and 150°. The sampling frequency of PDM was

also changed to 3.2 MHz corresponding to the sampling frequency of PCM of 100 kHzm and used different direction to record the dog whistle. The figure of system design is showed below.

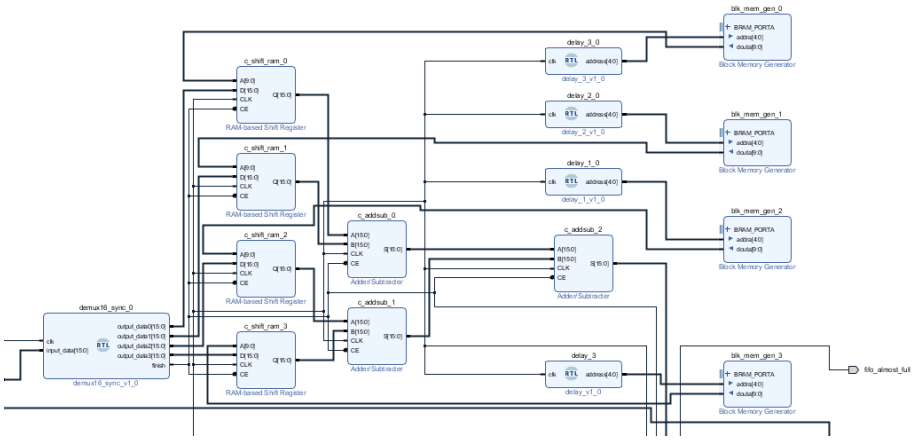

Figure 5.5: The partial design of system design of fixed several angles delay in vivado

5.5 Summation of multiple sensor inputs with scanning delays (120 angles totally from 30°to 150°)

Adding scanning block to control the delay lines in system design in vivado. There are 120 sets of delays corresponding to a sweep from 30°to 150°for four microphones, use enable_adder and scan_finish flag signal to control the energy calculation block for each angle. The figure of partial design of scaning and energy calculation is showed in fig 5.6 below.

The final energy data was imported in Matlab, and there were negative values in the result, which was unexpected. Attempting to output debug values through the FIFO generator, such as all zeroes, was also unsuccessful, suggesting an issue with the implementation.

The report in vivado also showed a large negative timing slack in the path of hardware after the implementation , indicating a latch in demux part. The demux code was modified to remove the latches and since it used combinatorial process, all the input signals in the sensitivity list were not included. However, another source of slack was due to the multiple adders (which was used to add the delayed input signals) which also resulted in layout problem(fanout). To remove the slack due to multiple adders, the code was written in VHDL and implemented as a single IP block. The larger negative slack became small eventually. The figure of timing slack is showed in fig 5.7

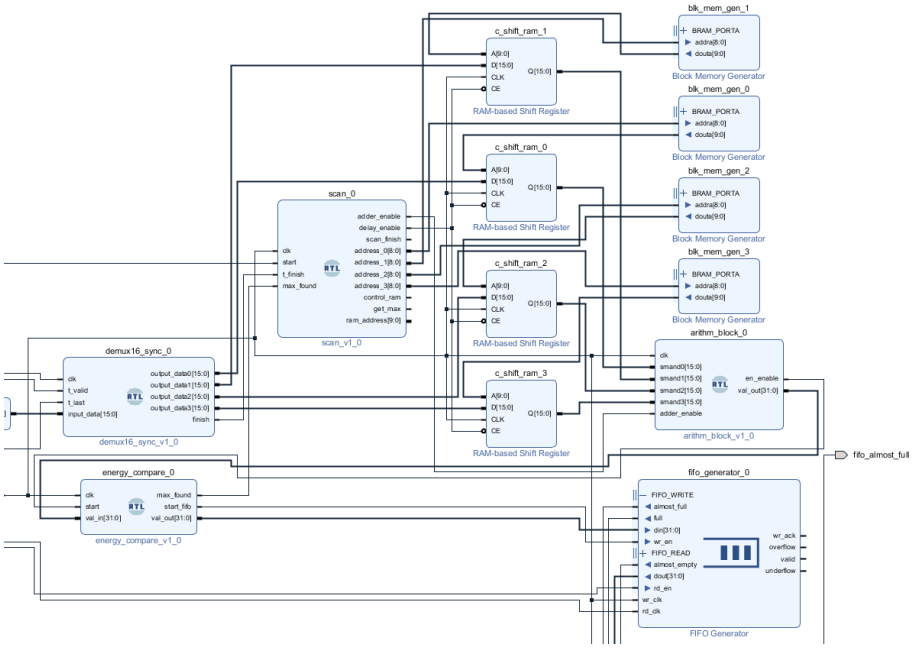

Figure 5.6: The partial of system design of scaning angle block in vivado

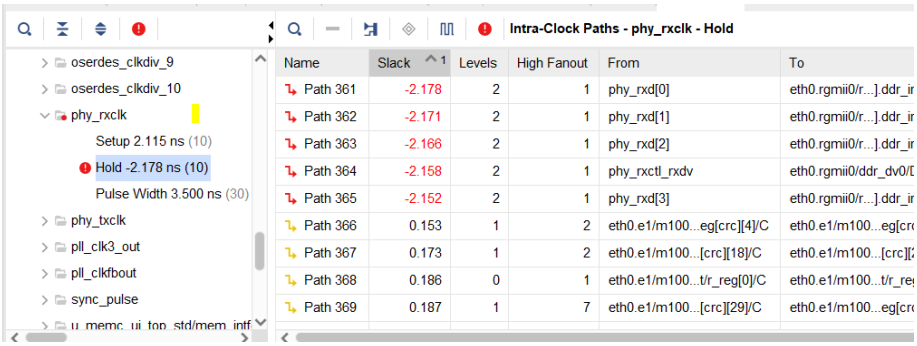

Figure 5.7: There is still small negative timing slack in vivado

5.6 Simulating System Functionality in Questasim

Designing iteratively in actual hardware is both inflexible and time-consuming, with sythesis runs often exceeding 15 minutes for the design described in this report. This is partly due to the fact that a significant amount of supporting hardware is needed to make it implementable and to interface with external devices.

This motivates the use of simulation to test for correct functionality under idealized conditions, without including the aforementioned supporting hardware. This allows much faster design iteration, as results can be generated in very little time, and can quickly eliminate issues that do not depend on the implementation itself.

This allows experimenting and gauging the effects of system parameters and different component implementations.

To this end, a *model* of the core functionality (delay-and-sum processing, multiplexing/demultiplexing, energy computation and angle-scanning) was constructed in VHDL and simulated in Questasim. Initally, a purely behavioural, non-synthesizable model was written with the goal of trying to produce results that match the theoretical results produced in MATLAB. This model was successively modified to exchange behavioural components for synthesizable counterparts.

Since the model does not have a real input signal, this too had to be simulated. The inputs, representing parallel data channels coming from the sensors, we simulated as pure, full-range sine waves with the same frequency as the ultrasonic audio source. These could be phase-offset, in order to simulate the effects of different incident angles

The end result was a model that is synthesizable (though perhaps not implementable due to timing requirements) that produces results that align with theoretical beam pattern plots, as seen in fig. 5.8.

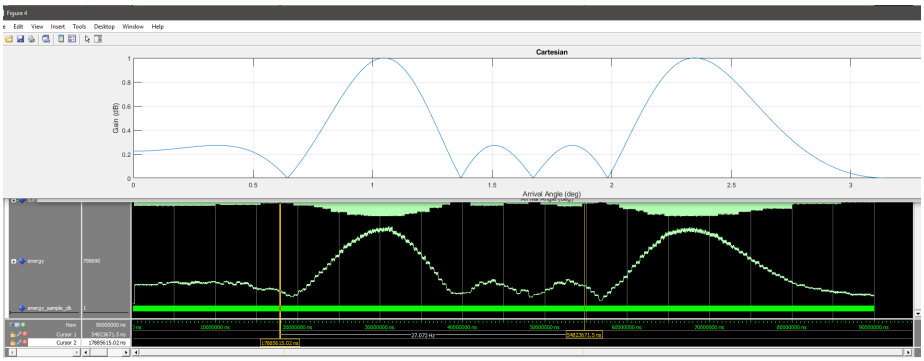

Figure 5.8: Questasim simulation of the HDL model of the core system. Theoretical MATLAB model overlaid for comparison. The HDL model adjusts its delay values to sweep between $[0, \pi]$ rad.

CHAPTER 6 Results

The following results were obtained :

- The delay values for each microphone for every angle within the range of 30°to 150°is calculated by using the MATLAB script.
- By using the multi-channel mode in the CIC and FIR filter blocks given in the reference design, audio was recorded with multiple microphones. We listened to the recorded audio using AUDACITY and confirmed that the signals were sampled properly.
- Implementing delay values of seven angles for four microphones by RAM-based shift register and ROMs, whcih can be selected to point to an adrress for a specific angle and test it in the different location of sound direction to see the sound signal resul in AUDACITY.
- For getting the derectional energy we have to change the delays to point the beam in the set of angles by implementing FSM to finish the sweep of delay values for each angle.
- Calculating energy of the audio for each angle and compare the muximum value to get the directional energy, dump it from board to analyze in Matlab.

6.1 Discussion and Analysis

Even though we get the delay values for all angles, spatial aliasing occurs at the extremes i.e, 0°and 180°. But with increasing number of microphones, spatial aliasing decreases as shown in figure 6.1.

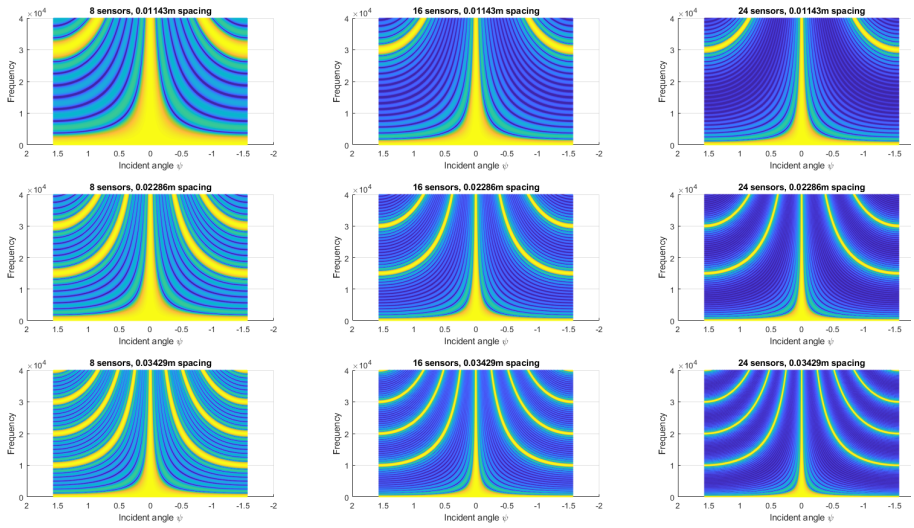

Figure 6.1: Spectral aliasing with varying number of microphones (horizontal axis) and increased spacing between adjacent sensors (vertical axis). Each plot shows frequency response vs. incident angle for 0 to 40 kHz.

As the FPGA hardware design was iterated upon throughout the project, the synthesized and implemented design continually failed to meet timing requirements. This was already apparent in the unmodified reference design, and the failing signal paths were traced to the ethernet controller, rather than anything implemented in this project. In practice, the design did not seem to be affected by this and so no attempts were made to correct it, as it was deemed out-of-scope. There is, however, the possibility that, as the design became more complex, this negative timing slack was exacerbated, leading to timing issues in other parts of the design. For any future development, the negative timing slack in the ethernet controller should be corrected. One possible way in which this might be achieved is through explicit floorplanning, as described in a Xilinx tutorial video [13]. This involves isolating critical parts of the design so that they meet the timing requirements and then "locking" their implementation in place. When subsequently implementing the design as a whole, the tool will take into account the "locked" elements and design around them. This may clearly result in sub-optimal placement of some parts of the design, but hopefully guarantee that all timing requirements are met.

CHAPTER 7 | Conclusion & Future Work

7.1 Conclusion

Although we initially aimed to achieve a fully functioning system to track the source, we were not able to use the data that was recorded for further analysis. But all those efforts were not in vain because we learned a lot about tackling the difficulties of designing a system in Vivado and ultrasonics, more specifically about the spectral aliasing and how it affects the design considerations. For a system to work in real time and be reliable, the most important factors are latency and accuracy which can be drasticlally improved in the future.

7.2 Future Experiments

7.2.1 Identify the reason for timing slack in the system design in vivado

There is still a small negative timing slack in the system design in vivado, hence the source of the slack must be found and fixed in the future, to think further about how to write the code to prioritise the final design and layout.

7.2.2 Verifying scaning 120 angles in ROMs to 24 microphones

Output signal should still be sinusoidal but amplitude should conform to spatial filtering function.

7.2.3 Computing approximate energy at each angle from summed signals

On FPGA or in MATLAB; might be better to do in MATLAB as the formula (at least in the naive, direct implementation) requires a lot of resources and very wide data words to maintain precision (precision that may not be needed)

Implementing beamforming (using shift registers etc) and:

- Manual verification of sound source at fixed angle using fixed delay-values.

Change angle manually using buttons.(get respective pre-computed delays from a source file for each increment/decrement in angle)

Potentially use LCD for debug information (angle of beam etc...)

- Sweeping along one dimension at fixed rate

  What factors will limit the sweep rate/refresh rate?

7.2.4 Extend to implement 2 microphone array

The system, as it is currently designed, only implements 4 microphones in the sensor array, though extending it to support all 24 microphones requires only minimal design effort due to the generic nature of the design. The same is largely true of implementing support for a second sensor array in order to scan along an orthoganol dimension.

Bibliography

[1] H.-U. Schnitzler, C. F. Moss, and A. Denzinger. "From spatial orientation to food acquisition in echolocating bats". In: *Trends in Ecology & Evolution* 18.8 (2003), pp. 386–394.

[2] A. Elfes. "Sonar-based real-world mapping and navigation". In: *IEEE Journal on Robotics and Automation* 3.3 (1987), pp. 249–265.

[3] A. A. M. Khalaf, A. Eldaly, and H. Hamed. "Different adaptive beamforming algorithms for performance investigation of smart antenna system". In: Sept. 2016. DOI: 10.1109/SOFTCOM.2016.7772134.

[4] J. Benesty, J. Chen, and Y. Huang. "Conventional Beamforming Techniques". In: *Microphone Array Signal Processing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 39–65. ISBN: 978-3-540-78612-2. DOI: 10.1007/978-3-540-78612-2_3. URL: https://doi.org/10.1007/978-3-540-78612-2_3.

[5] 2020. URL: http://www.labbookpages.co.uk/audio/beamforming/composite.html.

[6] D. Antonsson and M. Li. *Ultrasonic Source Localization with a Beamformer Implemented on an FPGA Using a High-density Microphone Array*. 2018. (Visited on 03/25/2020).

[7] *Field programmable gate array*. 2020. URL: https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html (visited on 03/24/2020).

[8] *FPGA fundamentals*. 2019. URL: https://www.ni.com/sv-se/innovations/white-papers/08/fpga-fundamentals.html (visited on 03/24/2020).

[9] S. HAUCK. "The Roles of FPGA's in Reprogrammable Systems". In: IEEE, 2019. DOI: 10.1109/5.663540. (Visited on 03/24/2020).

[10] *Top 10 FPGA advantages*. 2019. URL: https://hardwarebee.com/top-10-fpga-advantages/ (visited on 03/26/2020).

[11] *Product data sheet SPH0641LU4H-1*. 2015. URL: https://media.digikey.com/pdf/Data%5C%20Sheets/Knowles%5C%20Acoustics%5C%20PDFs/SPH0641LU4H-1.PDF (visited on 03/26/2020).

[12] *CIC Compiler v4.0 LogiCORE IP Product Guide Vivado Design Suite*. 2016. URL: https://www.xilinx.com/support/documentation/ip_documentation/cic_compiler/v4_0/pg140-cic-compiler.pdf (visited on 03/25/2020).

[13] *Design Analysis and Floorplanning with Vivado*. 2020. URL: https://www.xilinx.com/video/hardware/design-analysis-floorplanning-with-vivado.html (visited on 05/27/2020).