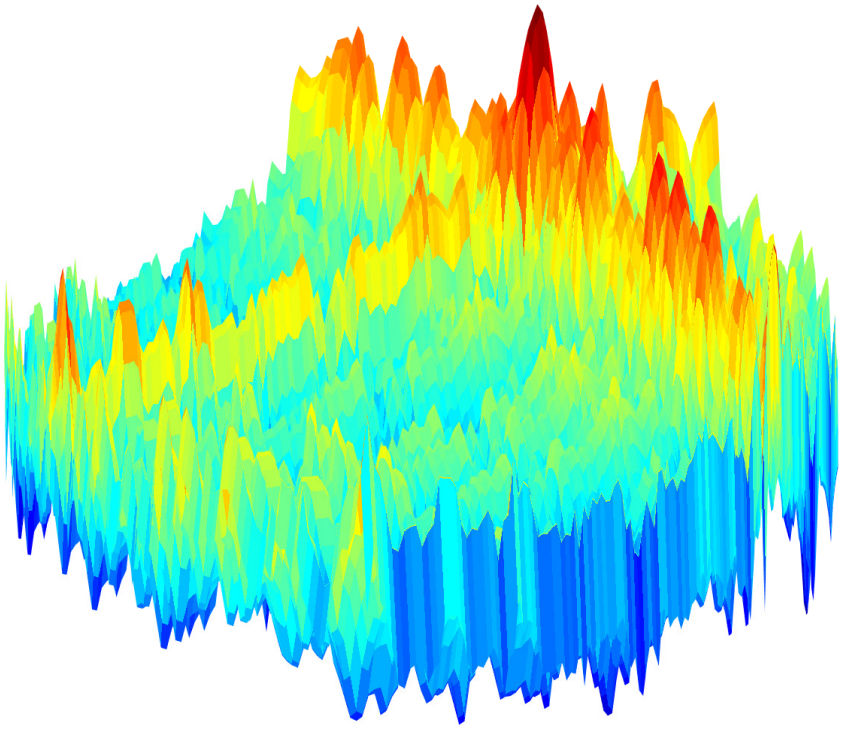


Ultrasonic Source Localization with a Beamformer Implemented on an FPGA Using a High-density Microphone Array

Master's thesis in Embedded Electronic System Design

Daniel Antonsson
Mengxuan Li

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Master's thesis 2018

Ultrasonic Source Localization with a Beamformer Implemented on an FPGA Using a High-density Microphone Array

Daniel Antonsson

Mengxuan Li

Department of Computer Science and Engineering
Programme of Embedded Electronic System Design
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

Ultrasonic Source Localization with a Beamformer Implemented on an FPGA Using a High-density Microphone Array
DANIEL ANTONSSON
MENGXUAN LI

Supervisor: Lena Peterson, Department of Computer Science and Engineering
Supervisor: Steinar Thorvaldsson, Syntronic
Examiner: Per Larsson-Edefors, Department of Computer Science and Engineering

Master's Thesis 2018
Department of Computer Science and Engineering
Programme of Embedded Electronic System Design
Chalmers University of Technology
University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Ultrasonic Source Localization with a Beamformer Implemented on an FPGA Using a High-density Microphone Array

DANIEL ANTONSSON
MENGXUAN LI
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

A system has been designed which uses a high-density MEMS microphone array and a beamforming algorithm implemented on an FPGA to perform real-time positioning of ultrasonic sources. The array consists of 48 microphones which is a limit derived from the latency of the system; if the latency were to increase further the system could no longer be said to operate in real time.

The system calculates signal energy in front of the array and displays the energy on a heatmap which updates at a frequency of 5.2 Hz. It can locate near-monochromatic sources at 20 kHz and narrowband sources at 35 kHz. However, the ability of the system to locate these sources was proven to be unreliable due to a phenomenon called spatial aliasing which causes an ambiguous pattern in the heatmap. Very brief sounds could not be located by the system and the system is also easily disturbed by ambient noise, such as noise from conversations.

It was concluded that the first step of any potential future development should be to completely revise the implementation of the beamforming algorithm in order to increase the amount of microphones that can be used and decrease the latency of the system.

Keywords: microphone array, ultrasonic, FPGA, beamforming, delay-and-sum

Acknowledgements

We would like to express our sincere and heartfelt thanks to our supervisors Steinar Thorvaldsson and Lena Peterson for providing excellent guidance during this thesis work.

We would also like to thank all the people working at Syntronic, especially Bhavishya Goel and Erik Ikdal, for providing us with resources which made our research possible and also for their helpful suggestions, valuable advice and detailed comments.

Daniel Antonsson & Mengxuan Li, Gothenburg, June 2018

Contents

Abbreviations

ADC	Analog-to-Digital Conversion
API	Application Programming Interface
BRAM	Block Random-Access Memory
BUFG	Buffer Global
CIC	Cascaded Integrator-Comb
DAS	Delay and Sum
DOA	Direction of Arrival
DPTI	Digilent Parallel Transfer Interface
DSP	Digital Signal Processor
DTFT	Discrete-Time Fourier Transforms
FF	Flip-Flop
FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
IO	Input/Output
IP	Internet Protocol
LUT	Lookup Table
LUTRAM	Lookup Table Random-Access Memory
MEMS	Microelectromechanical Systems
MMCM	Mixed/Mode Clock Manager
PC	Personal Computer
PCM	Pulse-Coded Modulation
PDM	Pulse-Density Modulation
RAM	Random-Access Memory
ROM	Read-Only Memory
SMD	Surface Mounted Devices
SNR	Signal-to-Noise Ratio
SRL	Shift Register Lookup Table
UART	Universal Asynchronous Receiver/Transmitter

1

Introduction

This is a project report that documents the work carried out in a master thesis project at Chalmers University of Technology in collaboration with the consulting company Syntronic. This chapter describes the background, purpose, problem description and limitation of the thesis work.

1.1 Microphone arrays and beamforming

In many fields, such as sonar, radar, wireless communications and seismology, there has been considerable research effort into sensor arrays [1], [2]. This is a group of sensors that is arranged in some geometrical pattern with the aim to enhance signals and separate them from noise [2]. For acoustic signals they are called microphone arrays.

Some applications in which microphone arrays have been found useful are separating sound sources to improve performance in hearing aids, security monitoring, studying wildlife, detecting air leaks and detecting small unmanned aerial vehicles [3]–[7]. Most of these applications depend on the array's ability to localize sound, which is the focus of this report.

Source localization is done by means of a technique called beamforming. A beamformer can be described as a spatial filter which enhances signals coming from a certain direction, the look direction, and attenuates signals originating from any other direction [8]. Conforming to this analogy: filters have passbands and stopbands and similarly beamformers have main lobes and side lobes. The main lobe is the high amplitude portion of the beamformer and the side lobes are the attenuated portions. The usefulness of this filtering property lies in steering the beam to various locations and determine which direction corresponds to the highest signal intensity - in our case: the loudest sound. How the beam is steered will be explained in Chapter 2. The ability of microphone arrays to accurately identify the correct position depends on the following factors [8]–[12]:

- Number of microphones
- Spacing between microphones
- Weighting of individual microphone gains

- The size of the array

A higher number of microphones makes the beam more narrow, which means that sound can be spatially filtered more precisely. Additionally, the signal-to-noise ratio (SNR) of the array increases as the number of microphones increases [13]. Less spacing between microphones prevents spatial aliasing [14], a phenomenon that is explained in Chapter 2, and it also increases the array's SNR [13]. More spacing between microphones, on the other hand, makes the beam narrower. The weighting of individual sensors, which may be considered as gain factors, affects the level and shape of the side lobes. Finally, the accuracy increases as the size of the array is increased [9].

In a nutshell: the array's ability to locate sound sources improves as the number of microphones is increased, the size of the array is increased and the density of microphones is increased. But the ability to decrease spacing between microphones is obviously limited by the size of the microphones and increasing the amount of microphones increases the amount of data to be processed.

The problem with size is somewhat mitigated by using microelectromechanical systems (MEMS) microphones; this technology has led to the development of small microphones with very high performance. However, processing large amounts of data from many sensors is likely not feasible on a regular computer processor, especially if the system is to operate in real time. Additionally, using a microphone array with a large amount of microphones means there will be a need to handle a large amount of input/output (I/O). Managing these problems can be made easier by implementing such a system on a Field-Programmable Gate Array (FPGA).

1.2 Field-Programmable Gate Array

An FPGA is a device which is comprised of programmable logic blocks and programmable routing and I/O blocks. They can be programmed with hardware description languages to achieve a vast variety of functions [15]. The most fundamental building blocks of FPGAs are registers, multiplexers (mux) and lookup tables (LUTs). Registers, also called flip-flops, are used to store one bit of data; muxes are used to select one of several input signals and forward the selected input to the output; LUTs can be seen as a multiplexer with fixed input values. FPGAs have programmable routing, which means wiring between logic blocks inside the FPGA can be arranged by the designer. FPGAs can have a high I/O count and each I/O pin can be configured as an input, output or bidirectional I/O [16].

Implementing the system on an FPGA opens up the possibility of an application specific architecture. For instance, if the system turns out to be too slow, the architecture can be designed to carry out all operations in parallel to the extent that resources permit it. Conversely, if the system has too high hardware requirements, it can be designed to carry out operations in series. Additionally, the MEMS mi-

crophones (from this point on referred to as sensors) have built-in analog-to-digital conversion (ADC) [17], which make them convenient to use with an FPGA.

The company Xilinx, the inventor of the FPGA [18], provides a software suite called Vivado for synthesis and analysis of designs made with hardware descriptive language (HDL). Besides many other features, this tool can also be used to program the FPGA with the synthesized design [19]. In this project Vivado has been used for all HDL design.

1.3 Research statement

The goal of this master thesis is to design a system that uses beamforming for real-time positioning of ultrasonic sources, using an FPGA platform and data that are collected by a high-density MEMS microphone array. As a metric of how successful the project is, the accuracy with which the system can localize a static source will be used. This, by extension, means that the system is to be designed to accommodate as many sensors in the array as reasonable within the FPGA resource constraints, since increasing the number of sensors improves the accuracy of the array. To maximize the chance for success in this project the system is to use an algorithm that is as simple as possible without critically compromising its accuracy.

1.4 Problem description

The section presents a more detailed analysis of what the goals asserted in Section 1.3 entail. Firstly, some considerations of processing ultrasound are explored, which is followed by the introduction of a simple algorithm and finally the ramifications of requiring operations to occur in real time are explored.

This project focuses on the problem of source localization for ultrasound, i.e. sound that is beyond the range of what is audible for humans. Focusing on ultrasound is interesting for many applications: air leaks in pressurized air systems, for instance, produce ultrasonic signals [6] and might be difficult to detect at lower frequencies; the sonar-like abilities of bats use ultrasound to prevent the sound from scattering [20] and thus only microphone arrays with the ability to process ultrasound can localize this sound. One consequence of working with ultrasound is that processing it inherently requires a higher sample rate compared to lower frequency sound which in turn puts higher requirements on the system hardware. Another problem, which is detailed in Chapter 2, is that a phenomenon called spatial aliasing negatively affects performance at higher frequencies.

As mentioned in Section 1.3 a decision to use the simplest algorithm was made. The simplest and most straightforward algorithm is the delay-and-sum (DAS) beamforming algorithm, which also poses the most demanding hardware requirements [21].

This means that upon scaling up the amount of microphones to be used in a system using the DAS algorithm, one can expect resource consumption to become prohibitively large and that some measures to lower resource consumption would have to be made. One such potential measure could be to use ring buffers to avoid unnecessary storage of calculated delay values [22].

Also mentioned in Section 1.3 is that one of the goals of the project was for the system to operate in real time. In this context real time means that the system must be able to respond quickly enough to an acoustic event, which may last less than a second. For instance, an acoustic camera - a device for visualizing sound sources - may be considered to operate in real time at 10 image frames per second [22]. The requirement of having a system operating in real time puts the following restrictions on the system:

- For every look direction a number of audio samples must be collected to determine the energy of the sound. The amount of audio samples collected for each angle must be kept to a minimum.
- The size of the array cannot be too large, since the bigger the array is the larger delays are required to perform the DAS algorithm.
- The number of locations the beam is steered to must be limited, since every new look direction will require some time for processing.

1.5 Limitations to mitigate risks

A significant risk in a project like this is potentially setting too ambitious goals, which is why we try to limit ourselves as much as possible. We therefore limit our focus on algorithms and only do comparison between algorithms using high-level simulation tools, specifically MATLAB. As previously mentioned we gravitate strongly towards an algorithm that is thought to maximize our chance of a successful FPGA implementation.

Projects like this may often fail due to uncontrollable factors, such as not having access to hardware that is a prerequisite to the system. Prior to the onset of this project the design of custom hardware is already finished which mitigated this risk. In retrospect this consideration is likely crucial as a design error in the hardware is in fact found and could be corrected in the nick of time.

Many research projects focus on broadband beamformers since the goal is often to process speech and there is a desire to have consistent beamforming over a span of frequencies. In this project we spend no time considering broadband beamformers. This can be justified by the fact that we have no interest in the quality of sound but rather only of the sound source position and thus we conclude that a narrowband beamformer (which are simpler to design [8]) would suffice.

1.6 Thesis outline

Chapter 2 reviews the basic theory relevant to the thesis work. It contains two sections that introduce beamforming and spatial aliasing.

Chapter 3 explains the design steps taken to achieve the goals stated in Section 1.3. It is divided into three parts: system architecture, design decisions and testing methodology.

Chapter 4 presents the results of our work. This section shows both results from simulations and tests conducted on actual hardware.

Chapter 5 discusses the significance of the results presented in Chapter 4. Suggestions for future improvements is also included in this section.

Chapter 6 lists the conclusions which are responding to Section 1.3.

2

Basic Delay-and-Sum Beamforming Theory

The purpose of this chapter is to base this project in a mathematically stable foundation. In the following sections, the theory behind the implementation in the thesis project is described briefly.

2.1 Beamforming

Originally developed in the fields of radar and sonar, beamforming is a long researched technique with a multitude of applications. Two interesting techniques are the ability to detect the direction of arrival (DOA) and enhancing a desired signal. A beamformer can be thought of as a spatial filter which grants a microphone array the ability to "listen" in a single direction, cancelling out sound coming from other directions.

The process of spatially filtering a signal can in fact be conceptualized as two subprocesses. The first consists of delaying signals from each microphone in a way so that the sum of the signals aligns in time when coming from a certain direction and thus making signals from that direction constructively interfere while signals coming from other direction destructively interfere. This process is illustrated in Fig. 2.1. The second process works by applying weight coefficients to the signals and then add them together. Synchronizing the signals steers the direction of the beam while applying weights controls the beamwidth of the main lobe and the attenuation of the side lobes [8]. In this project we have focused only on the synchronization process.

One beamforming technique is the DAS beamformer. It is the least complex algorithm to implement but is the most resource intensive [21]. This algorithm works by delaying the signal at each sensor. Given a linear array with uniform distance between each sensor the output of this algorithm from the direction of α is described as

$$g(a,t) = \frac{1}{N}\sum_{i=0}^{N-1} s(t - \frac{ida}{c}), \tag{2.1}$$

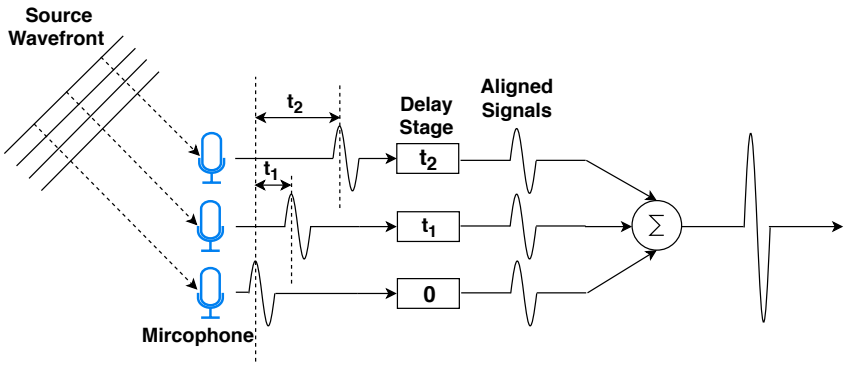


Figure 2.1: Block diagram of spatial filter.

where N is the number of sensors, s is the received signal, d is the distance between sensors, $a = \cos(\alpha)$, $\omega = 2\pi f$ is the angular frequency and c is the wave propagation speed. If we represent $s(t)$ by its continuous Fourier transform $S(\omega)$

$$s(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S(\omega)e^{j\omega t}d\omega$$

then

$$g(a,t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S(\omega)W(\frac{\omega a}{c})e^{j\omega t}d\omega,$$

where

$$W(\alpha) = \frac{1}{N}\sum_{n=0}^{N-1} e^{-j2\pi f n\cos(\alpha)d/c}. \tag{2.2}$$

$W(\alpha)$ is called the array pattern [23] and it is a plot of the array's response to a signal. In order to draw the array pattern we rewrite (2.2) as a function of ψ and θ, where ψ is a variable ranging $0^\circ \geq \psi \leq 180^\circ$ and θ is the look direction of the beam:

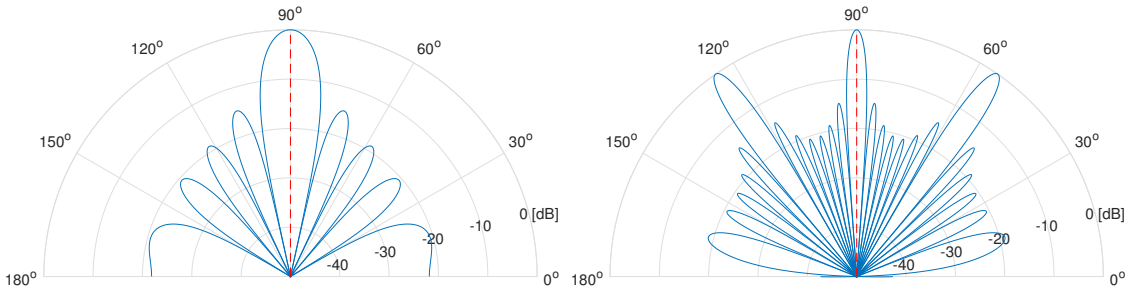
$$W(\psi,\theta) = \frac{1}{N}\sum_{n=0}^{N-1} e^{-j2\pi n f(\cos\psi-\cos\theta)d/c}, \tag{2.3}$$

the absolute of which can be shown to be

$$|W(\psi,\theta)| = \left|\frac{\sin\Big(N\pi f d(\cos\psi-\cos\theta)/c\Big)}{N\sin\Big(\pi f d(\cos\psi-\cos\theta)/c\Big)}\right| \tag{2.4}$$

as provided by [8].

The array pattern can then be plotted and the resulting figure can be seen in Fig. 2.2a. This gives us good insight into how the beamformer will spatially filter signals; it shows the width of the main lobe and the attenuation factor of the side lobes.



(a) **(b)**

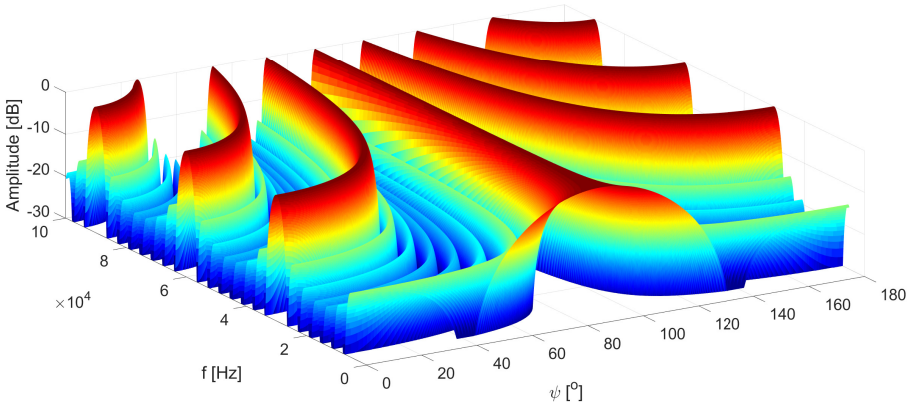
Figure 2.2: (a) shows an example beam pattern of an array with eight microphones, $f = 5\,\text{kHz}$, $d = 4\,\text{cm}$, $\theta = 90\,°$. The polar plot shows (2.4) plotted as a function of ψ. (b) shows and example of spatial aliasing of beam pattern of an array with eight microphones, $f = 15\,\text{kHz}$, $d = 4\,\text{cm}$ and $\psi = 90\,°$.

2.2 Spatial Aliasing

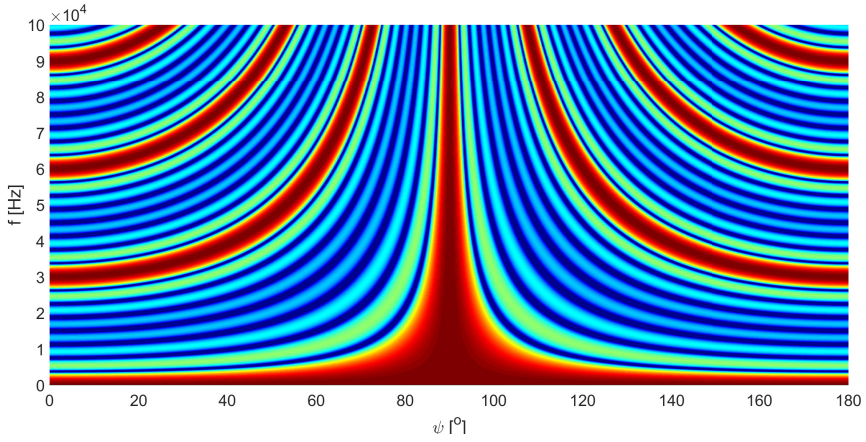
The Nyquist sampling theorem tells us that signals must be bandlimited to half the sampling rate to prevent aliasing from occurring. This phenomenon extends to signals that are functions of space coordinates and is then called spatial aliasing. The components created by spatial aliasing are called grating lobes [23]. If the spatial equivalent of the Nyquist criterion is not satisfied grating lobes will be produced, with the consequence that the array pattern will be ambiguous.

Fig. 2.2b shows an example of spatial aliasing occurring. Besides the main lobe at $\psi = 90\,°$ there are two grating lobes at $\psi = 55\,°$ and $\psi = 125\,°$. The relationship of the beamformer response versus the incident angle and frequency can be seen in Fig. 2.3, which displays a 3D plot and a heat map of the relationship. It is apparent that spatial aliasing becomes more prevalent for higher frequency signals.

It turns out, however, that spatial aliasing may not be as problematic as it can first appear. An intuition to this is that humans can localize sounds quite well even though an average spacing of 20 cm between the ears would suggest that spatial aliasing would occur for any sound above 850 Hz [24]. The patterns displayed in Figs. 2.2 and 2.3 were both generated using a monochrome source signal. The undesirable effects of spatial aliasing are somewhat mitigated when wideband signals are used [24]. This effect is illustrated in Fig. 2.4, where it can be seen that the two grating lobes shrink in amplitude as more (monochromatic) signals are added together and used as the source.



(a)



(b)

Figure 2.3: These plots show the relationship between frequency, incident angle and the beamformer response with eight sensors and $d = 11.43\,\text{mm}$ in the form of a 3D plot in (a) and a heat map in (b).

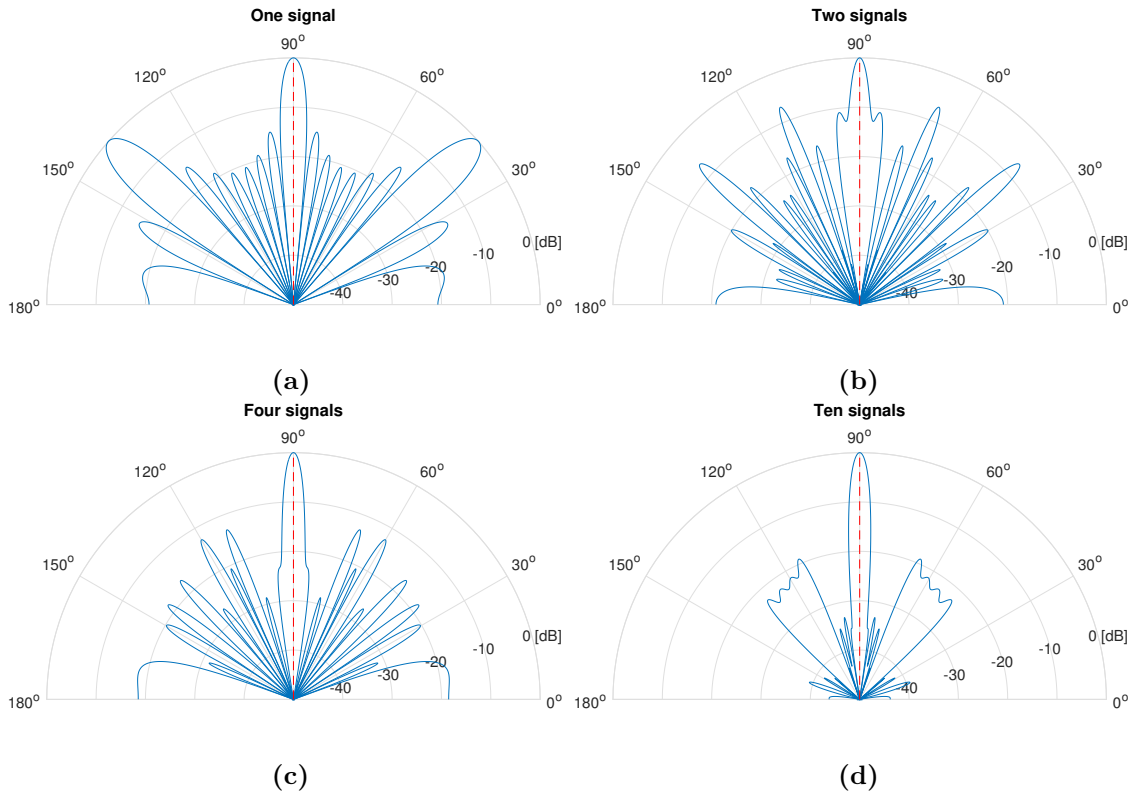


Figure 2.4: The figure depicts beamforming patterns of a system with eight sensors, $d = 11.43\,\text{mm}$ $\theta = 90\,^{\circ}$ and various input. In (a) $f = 40\,\text{kHz}$, in (b) the signal has two components of $f = 20\,\text{kHz}$ and $f = 80\,\text{kHz}$, in (c) the signal consists of four components ranging $20\,\text{kHz} \leq f \leq 80\,\text{kHz}$ and in (d) the signal consists of ten components ranging $20\,\text{kHz} \leq f \leq 80\,\text{kHz}$.

3

System Design Process and Testing Methodology

This chapter is meant to provide an overview of our system and document the system design process. Firstly the available hardware is presented in order to clearly show where some of the system's parameters originate from. Then the system architecture is explored in Section 3.2 which is divided into several subsections: audio processing, DAS beamforming, acquisition of directional energy, finite state machine (FSM), Digilent parallel transfer interface (DPTI), moving average in MATLAB and heat map. Section 3.3 contains explanations for any major design decision. Finally, our testing methodology is detailed.

3.1 Hardware

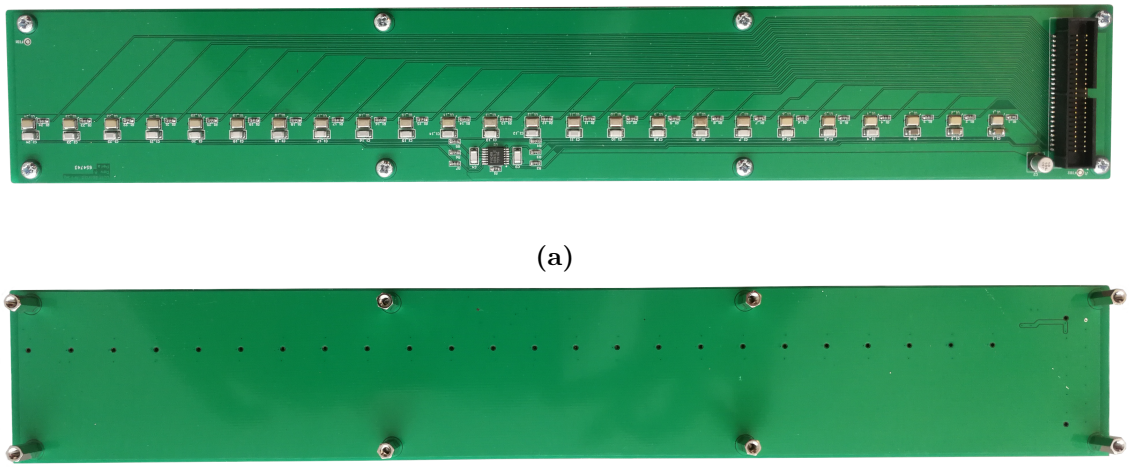
Some design parameters are derived directly from the available hardware. For that reason this section will present the available hardware.

3.1.1 Sensor Boards

The sensor boards (see Fig. 3.1) are custom designed by Syntronic and each houses 24 MEMS microphones [25] spaced 11.43 mm apart. This means that in this project we are constrained to this distance and our ability to vary the geometry of the array is limited to the configuration of the boards relative each other. In total we have six sensor boards available, which amounts to a total of 144 sensors.

3.1.2 Breakout Board

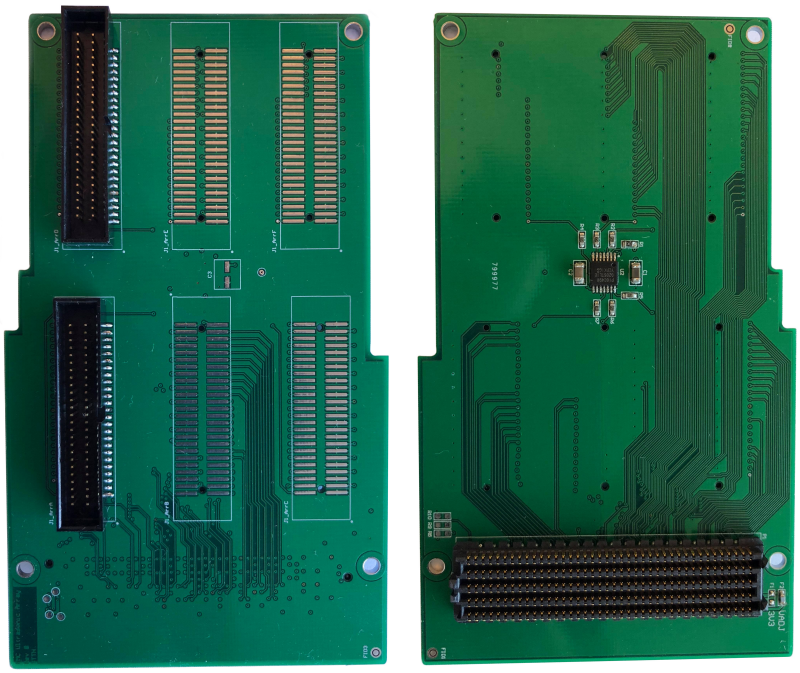
The breakout board (see Fig. 3.2) is custom designed by Syntronic to interface between the Genesys 2 Kintex 7 FPGA [26] development board and the sensor boards.



(a)

(b)

Figure 3.1: The front (a) and the back (b) of the sensor boards.



(a) **(b)**

Figure 3.2: The front (a) and the back (b) of the breakout board. (a) shows the SMD connectors on the front of the board that will connect the sensor boards to the breakout board. At the time of taking this picture only two connectors are soldered on the board. (b) shows a large SMD connector on the back which fits the FPGA Mezzanine Card connector on the Genesys 2 Kintex 7 FPGA development board.

3.1.3 Ultrasonic Transducer

Figure 3.3: The ultrasonic transducer.

The ultrasonic transducer (see Fig. 3.3) is made of a piezoelectric material which vibrates when an alternating voltage is applied to it. Its datasheet claims it has a center frequency of 40 kHz [27].

3.2 System Architecture

The system architecture overview is depicted in Fig. 3.4. It contains the FSM and the beamforming system. The beamforming system includes audio processing, DAS beamforming, acquisition of directional energy and interface.

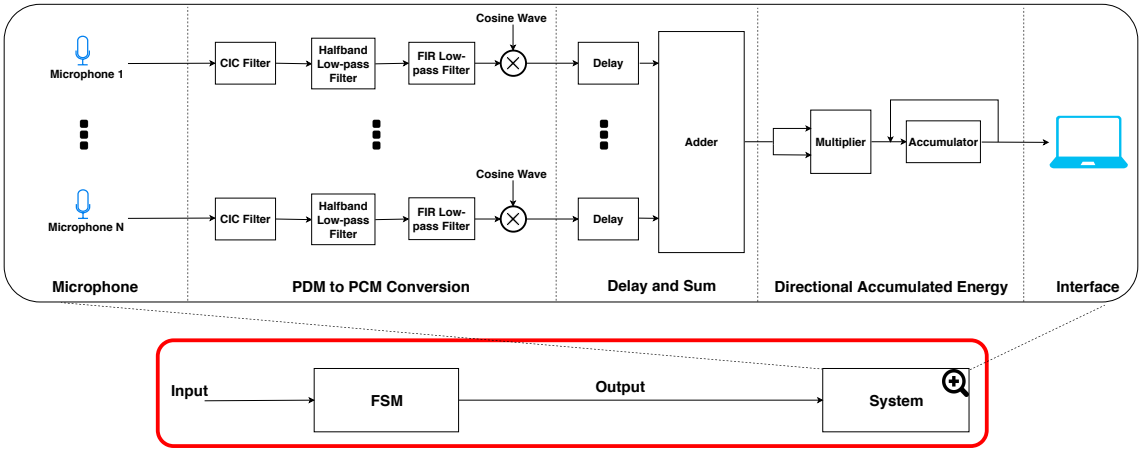


Figure 3.4: System overview.

3.2.1 Audio Processing

Conversion from pulse-density modulation (PDM) to pulse-coded modulation (PCM) is done on every individual data stream. An illustration of each component in this

conversion is shown in Fig. 3.5; as can be seen each one of these conversion blocks consists of several digital filters and a multiplier. The first filter is a cascaded integrator-comb (CIC) filter which is used to low pass filter the PDM stream in order to convert it into a PCM format; the PDM stream is essentially averaged. Then half band filters are used as compensation filters since the passband of CIC filters is not sufficiently flat [28]. The purpose of the finite impulse response (FIR) filter is to remove any high frequency noise left over from the conversion. Finally, the multiplier is used as a mixer to move the signal down to a lower frequency for making handling the signal easier. All of these components are implemented using Xilinx IPs.

The PDM clock is 4.608 MHz which is within the ultrasonic operation range according to the datasheet of the digital MEMS microphones [25] and a number evenly divisible by 192 kHz (a common sampling frequency).

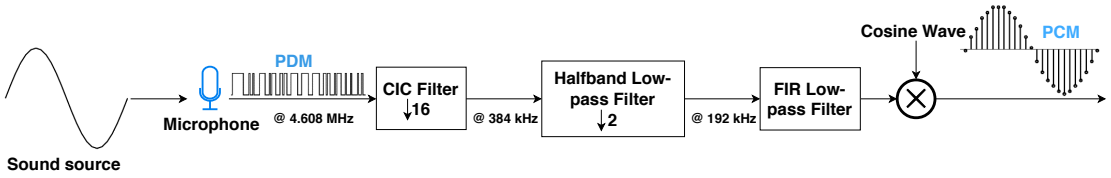


Figure 3.5: PDM to PCM conversion architecture.

3.2.2 Delay-and-sum beamforming

Delays are implemented with a Xilinx IP called RAM-based shift register. This IP has the option of variable length delays, which fits perfectly in this project since every scanning angle needs a different set of delays. The desired delay t_d is converted into a number of clock cycles $N_\mathrm{clk} = \mathrm{ceil}\,(t_\mathrm{d} f_\mathrm{clk})$, where f_clk is the frequency of the clock, and this number is driven to the input address of the shift register. A value for N_clk must be provided for each stream of input data and each scanning angle. These values are pre-calculated in MATLAB and saved into text files that are initialized as ROM upon compiling the system. A set of delays can then be chosen by pointing to an address in the ROMs containing the pre-calculated delay values which are then driven to the shift registers. Delay values for 128 different angles between 30 ° and 150 ° are pre-calculated in MATLAB.

Assuming speed of sound is 343 m/s, the largest possible delay using the shift register IP is 1024 clock cycles. With the PDM clock being 4.608 MHz this yields a maximum delay of $t_\mathrm{d} = 1024/4.608\,\mathrm{MHz} \approx 222\,\mu\mathrm{s}$. Assuming room temperature, and thus the speed of sound $c = 343\,\mathrm{m/s}$, the maximum needed delay for 24 sensors spaced $d = 11.43\,\mathrm{mm}$ apart is

$$t_\mathrm{max} = \frac{(N-1)d}{c}\cos(\theta_\mathrm{max/min}) = \frac{23 \cdot 11.43\,\mathrm{mm}}{343\,\mathrm{m/s}}\cos(30^o) \approx 664\,\mu\mathrm{s}, \tag{3.1}$$

for a system with maximum scanning angles of 30 ° and 150 °. This means that is necessary to either use three shift register IPs in a row or divide the clock that drives the shift register IPs by three. The current system uses the former solution, for reasons explained in Section 3.3.

The adding together of data streams is done with an adder IP which is controlled by enable signals from the FIR filters (which must traverse the shift registers). The mux in Fig. 3.6 selects the enable signal which is delayed the most to ensure that data from all the sensors are being added together and that there is no period of adding together data from just a few sensors. As the data need one clock cycle to be added together the enable signals are delayed one clock cycle for each stage of adders by being passed through a register, as shown in Fig. 3.6. This figure illustrates a nested loop which is defined by two constants, *width* and *depth*, that the designer decides. Arranging the adders in a nested loop like this allows for convenient changing of the number of sensors to be used. By generating an adder every time $i \bmod 2^{j+1}$ returns true for $0 < i < width$ and $0 < j < depth$ where $depth = \log 2(width)$ the formation shown in Fig. 3.6 is generated. Changing the number of sensors to, for instance, 16 would automatically add another layer of adders which decreases the amount of manual coding that has to be done.

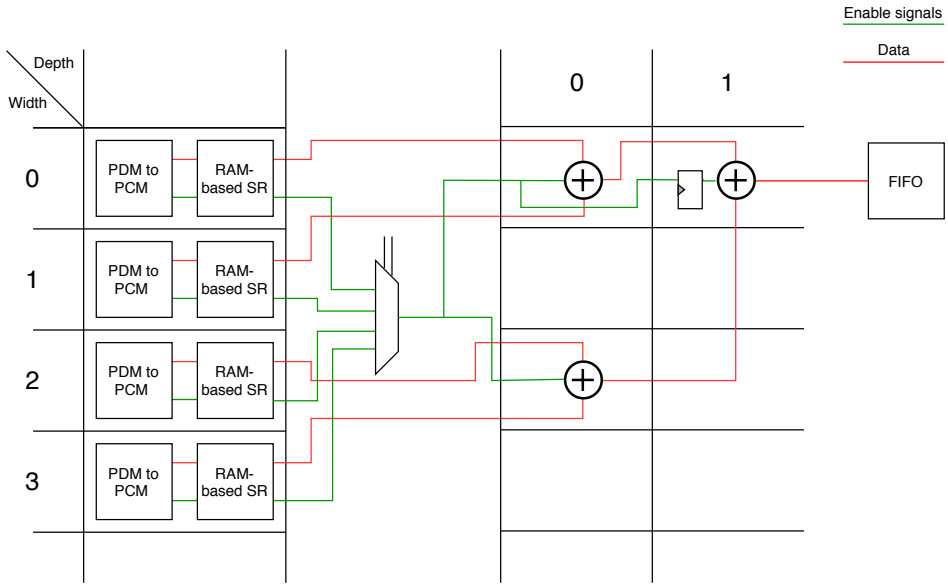
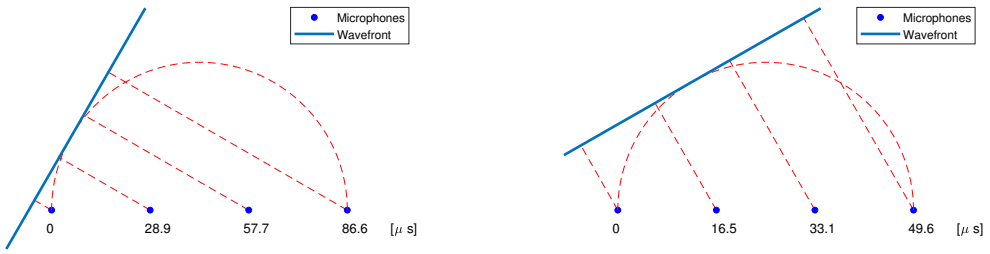


Figure 3.6: Four-microphone architecture. Clock signals are not shown to prevent cluttering. SR stands for shift register.

As earlier mentioned, delay values are pre-calculated. This is done using a custom designed MATLAB function which accepts as argument the number of sensors and the desired angles for scanning. For purposes of user feedback this function also creates an illustration of the wavefront incident on the microphone array, which can be seen in Fig. 3.7. This function calculates the delay values and converts them into a number of clock cycles, N_{clk}, which are written into plain text files that are subsequently used for initializing ROMs in the system.



(a) **(b)**

Figure 3.7: This figure illustrates how the delay values are calculated for a system using four sensors. The distance between microphones is hard coded to 11.43 mm. The value for the speed of sound used is $c = 343\,\mathrm{m/s}$.

3.2.3 Directional Energy

We would like to analyze the energy of the audio rather than just the raw audio. Therefore we need to do some processing in order to identify the energy of our summed audio data. The energy of a signal is calculated as

$$E_c = \int_{-\infty}^{+\infty} |x_c(t)|^2 dt,$$

where $x_c(t)$ is a continuous signal [29]. For discrete signals we have

$$E_d = \sum_{-\infty}^{+\infty} |x[n]|^2. \tag{3.2}$$

The physical implementation of this is straightforward and is depicted in Fig. 3.8. The Xilinx IP multiplier is used to square the output of the DAS beamformer. The accumulator adds together a number of samples to acquire the signal energy.

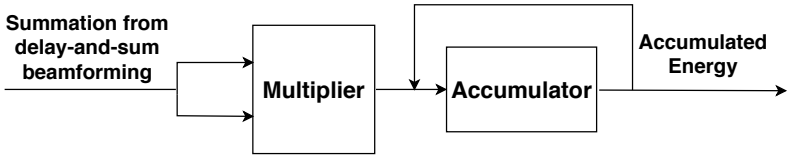


Figure 3.8: Acquiring directional energy as per (3.2).

3.2.4 Finite state machine

In the previous section we have created a beamformer with adjustable delays so that it can be steered to different angles. To get the directional energy we have to adjust the delays to point the beam in the desired angle, sample some data, and store it. Then we must do this for every angle we want to examine and then see which angle provides the largest energy, which will be the candidate for the direction of arrival (DOA). In order to do this we need a FSM, whose design is illustrated in Fig. 3.9.

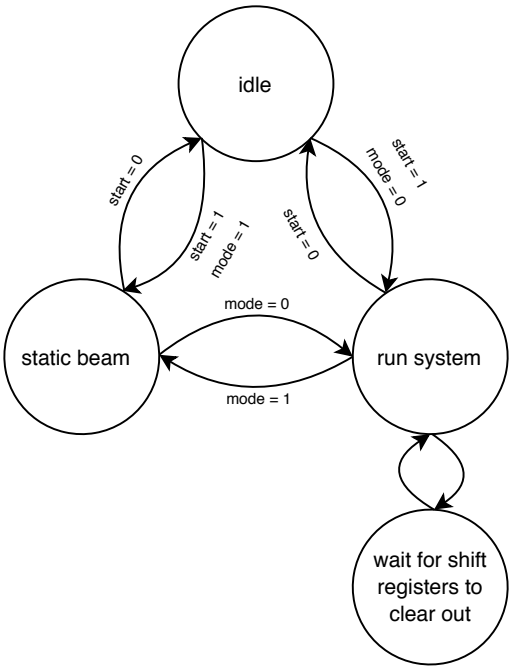


Figure 3.9: State diagram for the FSM. The input signals start and mode are set by the user via switches on the FPGA development board.

The *idle* state is for doing nothing while the start button is set to zero. The `run system` state is for gathering data samples and the `wait` state is for waiting until data from all sensors have propagated out of the shift registers. The delay values are provided to the shift registers by a ROM in which pre-calculated delay values are stored. Every time the system returns to the `run system` state a counter ticks up and a new set of data is read from the ROM. This setup is illustrated in Fig. 3.10. To get a better sense of the timing of the FSM Fig. 3.11 can be studied; any "Valid data" will occur during the *run system* state and the waiting periods in between occur during the `wait` state. Note that in Fig. 3.11 the length of the data vectors is still the same after applying the delay. This is because the shift registers use the SRL16/SRL32 mode of the slice LUTs which cannot be reset. Only the output registers of the IP block can be reset [30]. Because of this we have chosen to let any data still left in the shift registers propagate out before starting the delaying of a new set of data. In the actual system there are 128 sets of delays corresponding to a scan between 30 ° and 150 °, which means that the counter in Fig. 3.11 for the actual system needs to reach 127 before the entire sweep is finished. At that point

the FSM asserts a flag to indicate the finished sweep, which is helpful for verifying the frequency of the system.

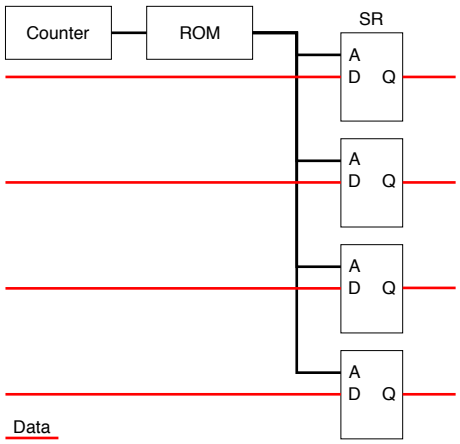


Figure 3.10: Illustration of how the delay lines are controlled. The ports on the shift registers are the address input A the data input D and the data output Q.

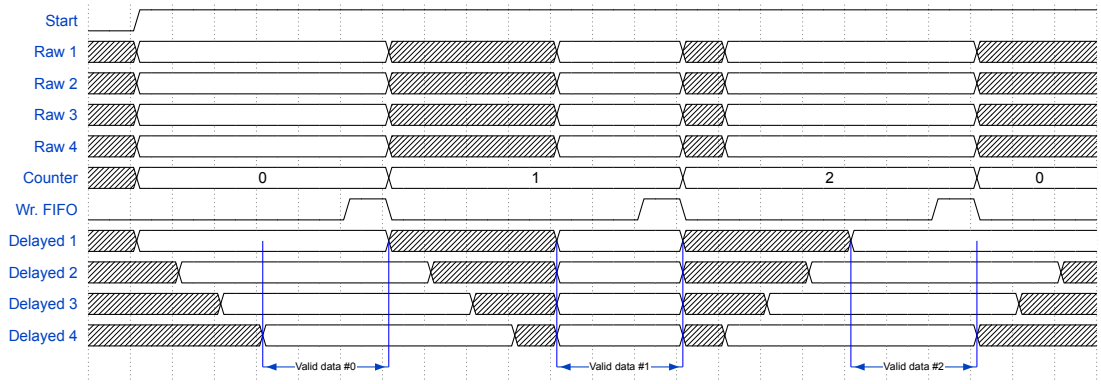


Figure 3.11: Timing diagram of the delay lines for a simplified version of the system which steers the beam to only three different angles. The signals Raw 1-4 represent raw data streams from the sensors. The signals Delayed 1-4 represent the data streams after they have passed through the shift registers. The signal Counter selects the set of delay values associated with Valid data #0, #1 or #2 and routes them to the address port of the shift registers. The signal Wr.FIFO tells the DPTI that data is ready for transfer (this data is a single bit vector which is attained after squaring and summing a whole data stream). In input A whole data streams are allowed to propagate out of the shift registers before starting a new set of delays.

The purpose for the `static beam` state is to fix the beamformer at 90° to enable verification of the system. In this state the system changes from transmitting an directional energy to the computer and instead transmits the output of the DAS beamformer, i.e. audio. This has enabled, for instance, the comparison between the system with and without the last FIR filter stage (implementing the system without the last FIR filter resulted in a large amount of noise and clipping and thus this filter was kept). This has also allowed for downloading samples of audio which has been useful for analyzing the spectrum of sound sources (see Fig. 4.4).

The total amount of time it takes for the system to finish a complete sweep can be calculated using Fig. 3.12 as a guideline: if $\tau_{\max}(i) =$ `Max delay`, $\tau_{\text{data}} =$ `Valid data` and $\tau_{\text{hold}}(i) =$ `Hold time` then

$$\tau_{\text{tot}} = \sum_{i=1}^{N} \tau_{\text{data}} + \tau_{\max}(i) + \tau_{\text{hold}}(i),$$

where τ_{tot} is the total delay, and N is the number of microphones. In the current system three shift register IPs are being used which means that the time required to empty them is divided by a factor of three, hence

$$\tau_{\text{tot}} = \sum_{i=1}^{N} \tau_{\text{data}} + \tau_{\max}(i) + \frac{\tau_{\text{hold}}(i)}{3}. \tag{3.3}$$

For the current system with $N = 24$ and delay values calculated for angles between 30 ° and 150 ° $\tau_{\text{tot}} = 96.2\,\text{ms}$, meaning that the frequency of the system is $f_{sys.} = \tau_{\text{tot}}^{-1} \approx 10.4\,\text{Hz}$.

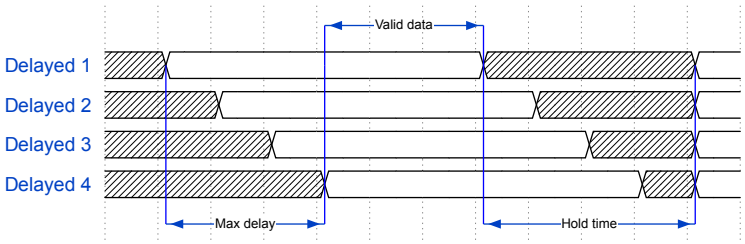


Figure 3.12: This figure depicts how much time the sampling of data from a single angle takes. *Max delay* represents the time it takes for sound to propagate across the whole array. *Valid data* represents the time it takes to sample data. *Hold time* represents the time it takes to let the shift registers clear out, at which point scanning of the next angle can begin.

Finally, the FSM also controls a flag that controls the multiplexing of the two board of microphones, the reason for which will be explained in Section 3.2.7.

3.2.5 Digilent Parallel Transfer Interface

For the FPGA used in this project an interfacing mechanism called DPTI is available which facilitates high-bandwidth communication between a host computer and the FPGA board. It contains two interfaces, of which one is the application programming interface (API) for the host, and another is the hardware interface on the FPGA. The DPTI subsystem uses an 8-bit bidirectional data bus and several control signals to achieve an asynchronous or synchronous parallel interface [31].

The timing diagram of the synchronous read mode that is used the system is shown in Fig. 3.13. In this mode, the transfer rate is 60 MHz and the maximum theoretical bandwidth is 480 Mbps [32].

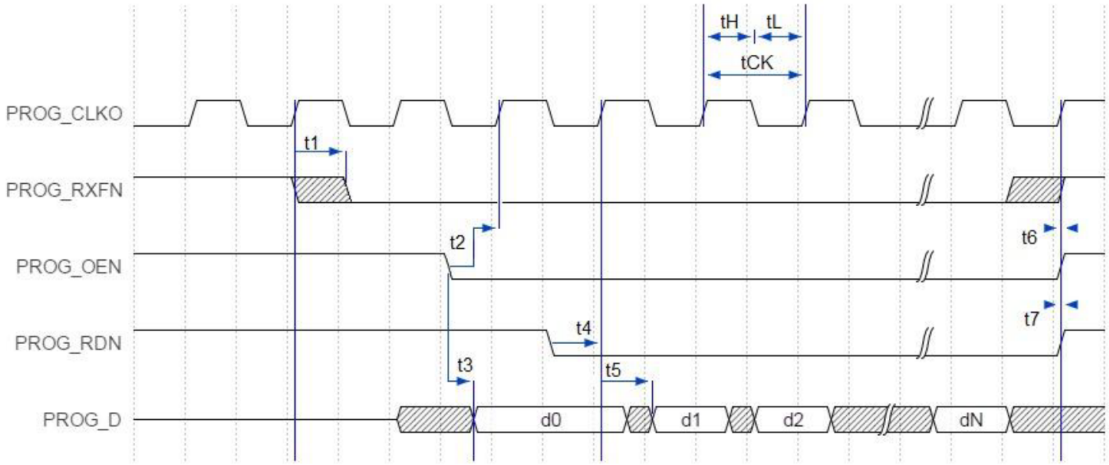


Figure 3.13: DPTI timing graph [32].

The synchronous read operation is started when PROG_RXFN is driven low by host. Then the PROG_OEN is driven low by the peripheral system to turn around the bus drivers before the PROG_RDN going low to acknowledge the data. After PROG_OEN goes low, the first byte of data is on the PROG_D bus. The data can be released from host by peripheral system when PROG_RDN is keeping low or the wait state is inserted in the PROG_RDN. If more data need to be read, the clock will be changed following PROG_RDN sampled low. When all data have been consumed, the PROG_RXFN will be driven high by the host. After PROG_RXFN is set to high, any data which appear on the data bus should be ignored [32].

The frame of input signal wr_di is shown in Table 3.1. Of this signal 24 bits are dedicated to the directional energy data, one bit is dedicated for the `Board Flag` and 7 bits are dedicated for the angle index.

Steered Response Power																							Board Flag	Index						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 3.1: The frame of wr_di.

The protocol is initiated through the API in the host computer and terminates when the user-specified number of bytes has been transferred. Currently the system scans 128 angles which thus results in 128 directional energy values, each of which is a 32-bit word. This means that transferring $128 \cdot 32$ bits $= 4096$ bits or 512 bytes will ensure a complete set of directional energy data, but it is not known at which point in the scan the transfer started. This is why the seven least significant bits ($\log_2(128) = 7$) of wr_di are used for providing the information about which angle the directional energy values correspond to. In the same fashion it is also not known whether the transferred data is coming from the horizontal or the vertical board, which is why another bit, `Board Flag`, is dedicated to provide this information.

Thus the eight least significant bit of the 32-bit directional energy values are being discarded.

3.2.6 Reading the Transferred Data and Calculating the Moving Average

The API used to control the DPTI is written in C++, but we wanted to use MATLAB for processing the data in order to get access to a multitude of MATLAB functions. The currently implemented solution is that the DPTI API continuously writes the directional energy data to a bit file and MATLAB continuously reads from the same file as long as there is a complete set of data from a sweep. Upon activating the MATLAB script it uses file position indicators to make sure the most recent data is being read from the file.

Once ported into MATLAB each complete set of directional energy data is averaged with a few of the previous sets to reduce transient behavior. Averaging just two sets showed significantly reduced flickering in the subsequent heatmap image. When averaging more than five sets of data, however, the image started to become sluggish; it would take approximately a second for it to react to changes in sound source locations. The MATLAB script has two modes: one which displays the two DOAs and one which displays a heatmap. If the mode for two DOAs is selected then after averaging, data from the two sensor boards are separated using the `Board Flag` and displayed in polar plots. If the mode for displaying a heatmap is selected, the data from the two sensor boards are separated and then combined in the fashion shown in Section 3.2.7. The combined directional energy data is mapped onto a circle as in Fig. 3.16b and plotted using the MATLAB function `surf`. This function allows for displaying both a heatmap and a 3D plot, both of which are displayed in chapter 4.

Finally, the MATLAB function `max` is used for picking out the largest value of the combined directional energy data, which is the candidate for the source. The corresponding angles are then displayed on the axes of the heatmap.

3.2.7 Heat Map

In order to expand the system from calculating a DOA to calculating a position two boards arranged perpendicularly to each other are needed, as per the arrangement in Fig. 3.14a. Sensors from two array boards are multiplexed so that they take turn in going through the processes described in Sections 3.2.1, 3.2.2, 3.2.3 and 3.2.5. This expansion can be done with virtually no increased resource consumption, albeit with the trade-off that the number of locations per seconds is halved which, according to (3.3), yields a frequency of $f_{sys} = 5.2\,\text{Hz}$. The boards are arranged in the geometrical constellation shown in Fig. 3.14a.

Since the DPTI (as it is currently used) starts at arbitrary points in the system's

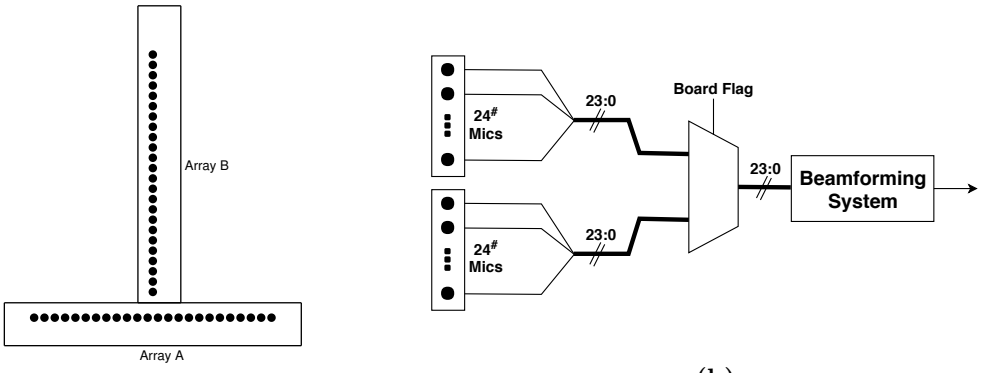


Figure 3.14: (a) shows the geometrical configuration of the arrays and (b) shows the multiplexing of the two arrays shown in (a).

execution it is necessary to dedicate a bit of the transferred data to indicate whether the data are associated with the horizontal or the vertical board, as earlier explained in Section 3.2.5 and as can be seen in Fig. 3.1. With this information it is now possible to "demux" the data from the two arrays which results in two vectors of directional energy data, which correspond to θ and ϕ of the spherical coordinate system. This means that the positions being scanned are in a "cone" with its vertex at the sensor array's center as seen in Fig. 3.15a.

To go from the aforementioned two 1-dimensional vectors to a 2-dimensional matrix the outer product of the vectors is calculated:

$$\mathbf{C} = \mathbf{b} \otimes \mathbf{a} = \mathbf{b}^T \times \mathbf{a}, \tag{3.4}$$

where $\mathbf{a}$ and $\mathbf{b}$ are 1×128 vectors representing directional energy data from sensor array A and B (see Fig. 3.14) and $\mathbf{C}$ is a 128×128 matrix which represents the 2-dimensional directional energy. A way to conceptualize how this works is to put some idealized data in (3.4) and see what happens. Say, for the sake of simplicity, that five angles per sensor array are to be scanned and that there is a single sound source somewhere in front of the arrays:

$$\mathbf{a} = \begin{bmatrix} 1 & 1 & 10 & 1 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 10 & 1 & 1 & 1 & 1 \end{bmatrix}$$

and

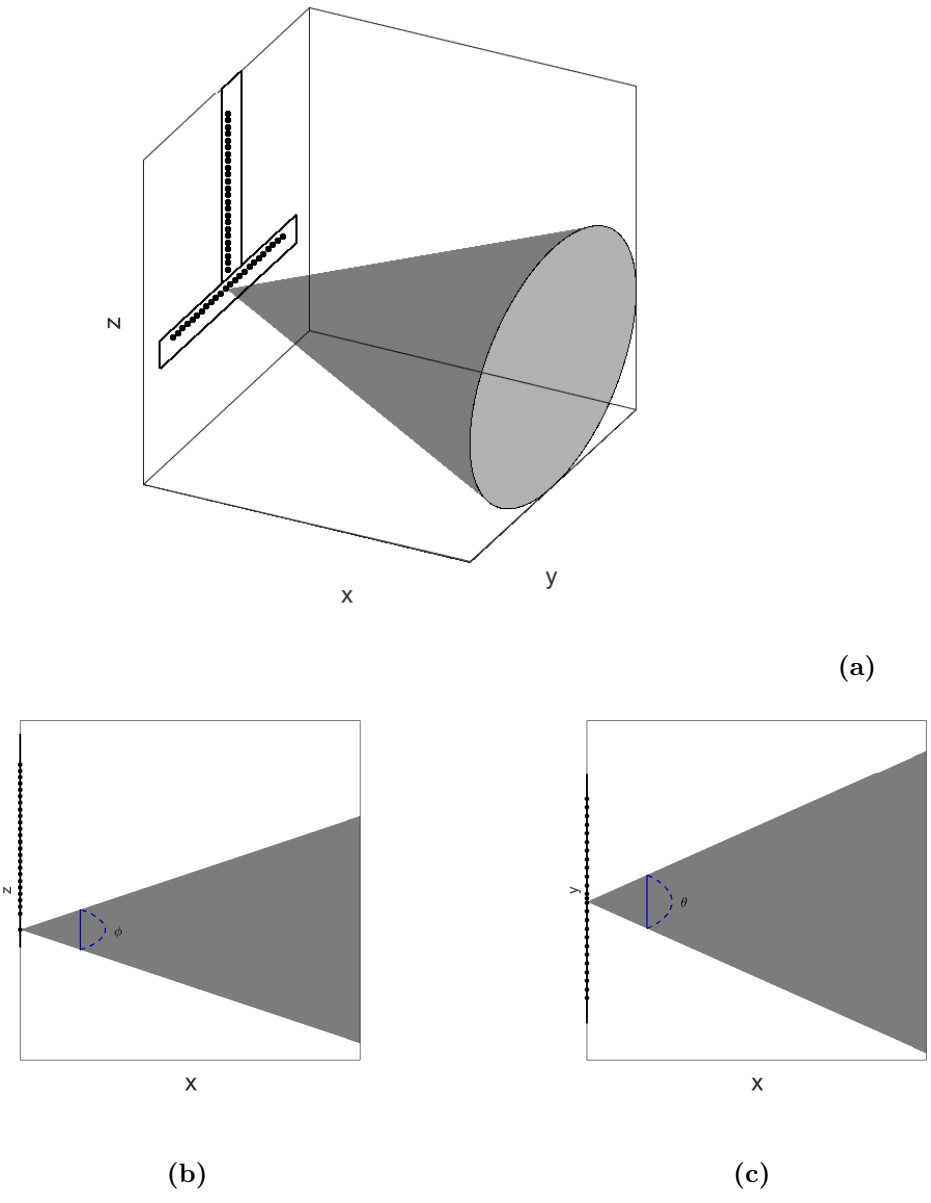
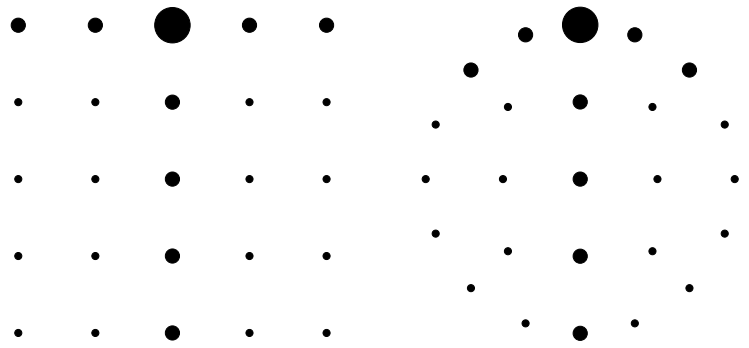


Figure 3.15: This figure illustrates the region in front of the array which is being scanned. In (a) the array and its scanning "cone" can be seen at an angle. In (b) the array and the cone can be seen from the side and in (c) the array and the cone can be seen from above. Note that this is just an illustration and in actuality the angles θ and ϕ are much larger.

$$\mathbf{C} = \begin{bmatrix} 10 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 10 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 10 & 100 & 10 & 10 \\ 1 & 1 & 10 & 1 & 1 \\ 1 & 1 & 10 & 1 & 1 \\ 1 & 1 & 10 & 1 & 1 \\ 1 & 1 & 10 & 1 & 1 \end{bmatrix}. \tag{3.5}$$

The horizontal vector, **a**, has a large directional energy value in its middlemost element, which represents a sound source at $\theta = 90\,^{\circ}$; the vertical vector, **b**, has a large directional energy value in its leftmost element, which represents a sound source at the array's smallest angle ($\phi = 30\,^{\circ}$ for this system). When these matrices are multiplied as in (3.4) it results in a combined set of directional energy values as in (3.5). Finally, the **C** array can be used to visualize the combined beamformers. One such visualization can be seen in Fig. 3.16, where the size of the dots corresponds to the directional energy values. As seen in Fig. 3.15a the scanning area is in actuality a circular disk, so to truthfully visualize the 2-dimensional directional energy values from array **C** must be mapped onto a disk as seen illustrated in Fig. 3.16b.



(a) **(b)**

Figure 3.16: This figure illustrates the results of combining the two perpendicular beamformer. In (a) the directional energy values are straightforwardly mapped on a square. In (b) they are mapped onto a circular disk, which represents a cross section of the combined beamformers.

Before ending this section a comment should be made on the seemingly arbitrary geometrical configuration of the sensor boards illustrated in Fig. 3.14a. Conceivably, it would be more desirable to have an entirely symmetrical configuration, but it is not possible to create a symmetrical configuration with two sensor boards perpendicular to each other without blocking some sensors of one of the boards.

3.3 Design Decisions

This section is dedicated to explaining the various major design decisions which were not obvious choices.

Choosing the DAS algorithm

As asserted in this project's research statement we want to limit the complexity of this project in order to maximize our chances of successfully implementing a working prototype. This was the main factor in our decision to base the system on the DAS beamformer.

Limiting the scanning angles between $\theta = 30\,^{\circ}$ and $\theta = 150\,^{\circ}$

As mentioned in Section 3.2.2 the system only scans angles between $\theta = 30\,^{\circ}$ and $\theta = 150\,^{\circ}$. There are two reasons for designing the system this way. The first reason is that the beamformer's main lobe becomes less and less sharp as it approaches $0\,^{\circ}$ or $180\,^{\circ}$. This can be observed in the simulation results in Section 4.1 and in particular Fig. 4.2. This would have as a consequence that the system would not be able to distinguish between angles close to $\theta = 0\,^{\circ}$ and $\theta = 180\,^{\circ}$. The second reason is that the maximum delay needed for the DAS algorithm increases depending on how close to $\theta = 0\,^{\circ}$ or $\theta = 180\,^{\circ}$ the beamformer is steered. This can be seen in (3.1) which, assuming θ is the only variable, will obviously be maximized for $\theta = 0\,^{\circ}$ or $\theta = 180\,^{\circ}$. A larger maximum delay means that the system frequency will be lower, which provides further incentive to limit the scanning angles.

Sending additional information besides directional energy over the DPTI

As explained in Section 3.2.5 the eight least significant bits of the directional energy values are being sacrificed to transfer one bit for indicating which sensor board is being used and which angle each directional energy value belongs to. The reason for this decision is simply because it was simple to implement. A more elegantly designed system would, for instance, have the DPTI send a start signal to the FSM which would then mean that the first transferred word would be known to correspond to whichever angle and sensor board that the FSM starts at, which would allow for using all 32 bits for the directional energy values. This was not done, however, as the development time of the DPTI became too time consuming.

The choice to use DPTI

DPTI provides a higher bandwidth transfer of data compared to, e.g., UART. But the system transfers 4096 bits every 92.6 ms as seen in (3.3) which means that its transfer rate is only $4096\,\text{Bytes}/92.6\,\text{ms} \approx 44.2\,\text{kbps}$, which can be comfortably handled by the UART protocol [26]. The reason for choosing the higher-rate protocol, DPTI, is to allow us to download and analyze data that have undergone only parts of the algorithm or potentially even raw audio from the microphones. Both of these actions require the transfer of much larger amounts of data and to keep options open the decision was made in favor of DPTI.

The number of scanning angles

The system's ability to differentiate between scanning angles depends on the shortest amount of time it can delay the data streams. With some trivial trigonometry it can be shown that

$$\Delta\theta = \arcsin(\frac{c\tau_{\min}}{d}), \tag{3.6}$$

where c is the speed of sound, d is the distance between sensors and $\tau_{\min}$ is the minimum delay. As mentioned in Section 3.2.2 there are three shift registers in a row which means that $\tau_{\min} = 3 \cdot f_{\text{clk}}^{-1} = 3 \cdot (4.608\,\text{MHz})^{-1} \approx 651\,\text{ns}$. This with $d = 11.42\,\text{mm}$ and $c = 343\,\text{m/s}$ (3.6) yields $\Delta\theta \approx 1.12\,^{\text{o}}$. This means that in the angles between $\theta = 30\,^{\text{o}}$ and $\theta = 150\,^{\text{o}}$ it should be possible to distinguish between $(150 - 30)/1.12 \approx 107$ angles. 107 values require seven bits to be represented in the wr_di - bus, so the decision was made to use all the 128 values that can be represented by seven bits. The strongest incentive for this decision was to increase the resolution of the scan (every angle can be thought of as a pixel in an image). Using slightly more than 107 values can be rationalized by remembering that only adjacent sensors may have identical delay values and sensors spaced further apart may not, so the DAS algorithm should by no means fail because of this. With 128 angles in the interval $30\,^{\text{o}} \geq \theta \leq 150\,^{\text{o}}$ there will be $120\,^{\text{o}}/128 = 0.9375\,^{\text{o}}$ between each angle. This would fundamentally be the smallest possible angle difference for the system to distinguish, but as indicated in (3.6) the system may be unable to distinguish between adjacent discrete angles.

Three shift register IPs in a row

As mentioned in Section 3.2.2 the current system uses three shift register IPs in a row to achieve the desired delay. The most obvious alternative to doing this would be to divide the clock by a factor of three or more and use only one shift register IP, which was how the system was originally designed. There are two disadvantages with this approach. The first disadvantage is that clock domain crossing requires a measure to prevent metastability [33] in data, such as using FIFOs. The second disadvantage is that clocking the shift registers with a slower clock will mean that the minimum delay, $\tau_{\min}$, will increase. The main incentive for using three shift register IPs in a row was to achieve a lower $\tau_{\min}$ and thus increase the resolution of the directional energy sweep. Unfortunately, the design is incomplete in the sense that currently the delayed data must pass through all three shift registers, which causes $\tau_{\min}$ to be three clock cycles. The reason this is not trivial to correct is because the MATLAB script that calculates the delay values was designed for one shift register IP and would have to be fundamentally redesigned for this problem to be solved. Additionally, the FSM would have to be modified to control the data streams so that they are able to pass through only one shift register IP and bypass the other two.

Multiplexing of whole system

The obvious benefit of this is that the system could be expanded upon to relay two DOAs instead of just one and that it could use twice the amount of microphones, virtually without consuming any additional resources. The obvious drawback to this is that the frequency of the system is halved, as mentioned in Section 3.2.7. Another major advantage with the decision to do this, however, was the simplicity of implementing it.

3.4 Testing Methodology

The testing methodology is based on the sound sources we have available, which is any sound source that can be produced by regular consumer speakers, i.e. anything below 22 kHz, and a piezoelectric ultrasonic transducer which has a center frequency at 40 kHz. A square wave of 40 kHz, the center frequency of according to its datasheet [27], is applied to the pins of the ultrasonic transducer. This produces an ultrasonic sound which can be used for testing the system's ability to localize such sounds. The `Static Beam` mode will be used to analyze the spectrum of these sound sources to ensure that the simulations and the system tests are performed with similar sound sources.

The first category of tests will be to have a static source in front of the system using one board with 24 sensors and thus only detecting the horizontal DOA. The second category of tests will all be done on the system that uses two boards with a total of 48 sensors. The results of all these tests will be presented in a heatmap. The tests will include:

- Testing with a single static source at 20 kHz
- Testing multiple static sources at 20 kHz
- Testing with a single static source at ultrasonic frequency
- Testing the system's performance while varying the distance to the arrays
- Testing how well the system can detect very brief sounds
- Testing how well the system can detect sounds in a noisy environment
- Testing how sensitive the system is to small movements of the sound source

4

Results

In this chapter, the project results are presented along with some observations. First the simulation results are reviewed which shows what to expect out of the implemented system and, subsequently, the implementation results are reviewed.

4.1 Simulation results

Fig. 4.1 shows the results of plotting (2.3) in MATLAB while to some extent attempting to replicate the sound source we have available. The frequency spectrum of the two different sound sources used in the tests can be seen in Fig. 4.4. The sub-ultrasonic sound source, seen in Fig. 4.4b appears well-behaved and can thus be simulated using a monochromatic source. The ultrasonic sound source seen in Fig. 4.4a appears to have several harmonics and we have attempted to replicate this spectrum in simulations. Specifically, the five peaks with the most power were selected and a signal with power equally distributed on these five frequencies was created for the simulation.

As predicted the sharpness of the beamformer is increased as the number of sensors is increased. It should be noted that the grating lobes have a slightly decreased amplitude, which is predicted in Section 2.2.

Another simulation was performed to investigate what happens at angles close to $\theta = 180\,^{\circ}$ (and thus also $\theta = 0\,^{\circ}$ since the patterns are symmetrical). The results can be seen in Fig. 4.2 and it should be noted that the width of the main lobe increases significantly as the beamformer is steered towards $\theta = 180\,^{\circ}$.

4.2 Implementation results

Currently the largest number of sensors we have been able to use is **48 sensors**, which is all the sensors on two sensor boards. The implementation result report from Vivado indicates that we will not be able to add many more sensors. An overview of the resource consumption can be seen in Table 4.3, which shows a large consumption

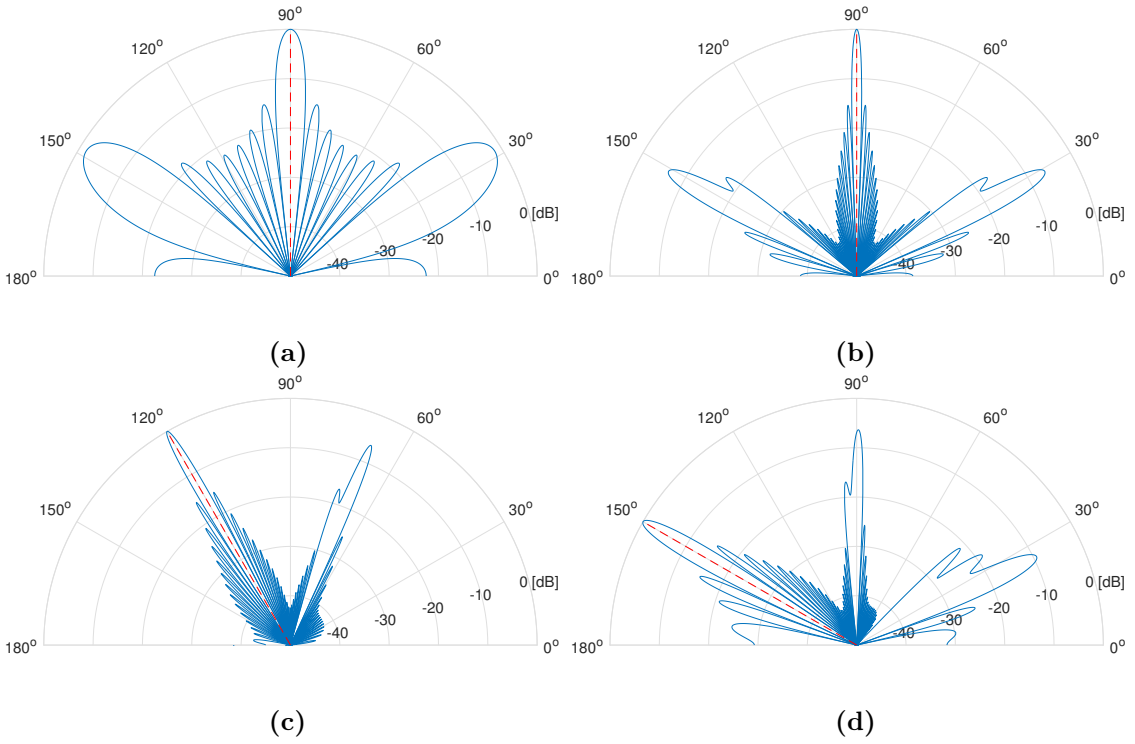


Figure 4.1: Simulation of our system with parameters decided by our hardware; $f = 40\,\text{kHz}$, $d = 11.43\,\text{mm}$ and $\theta = 90\,°$. The number of sensors is 8 in (a) and 24 in (b). In (c) and (d) the system with 24 sensors, with $\theta = 120\,°$ and $\theta = 150\,°$ respectively, are plotted. The sound source is modeled after the ultrasonic transducer.

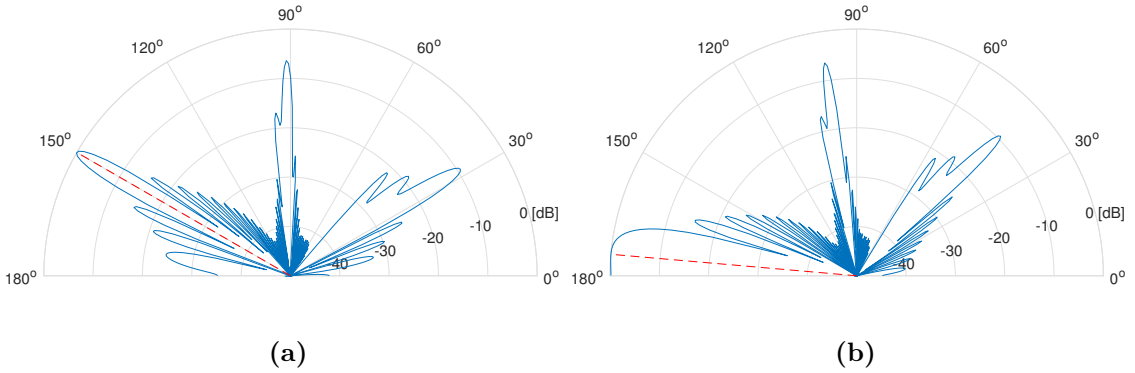


Figure 4.2: Simulation of our system with 24 sensors, $f = 34\,\text{kHz}$, $d = 11.43\,\text{mm}$ and $\theta = 150\,°$ in (a) and $\theta = 175\,°$ in (b).

of FPGAs LUTRAM. As mentioned already in Section 3.2.4 the system relays one set of directional energy data at a frequency of $f_{sys} = 10.4\,\text{Hz}$ when one sensor board of 24 sensors is used. When two boards, with a total of 48 sensors, are used and the two directional energy vectors are combined in MATLAB the resulting updating frequency of the heatmap image is, as mentioned in Section 3.2.7, $f = 5.2\,\text{Hz}$.

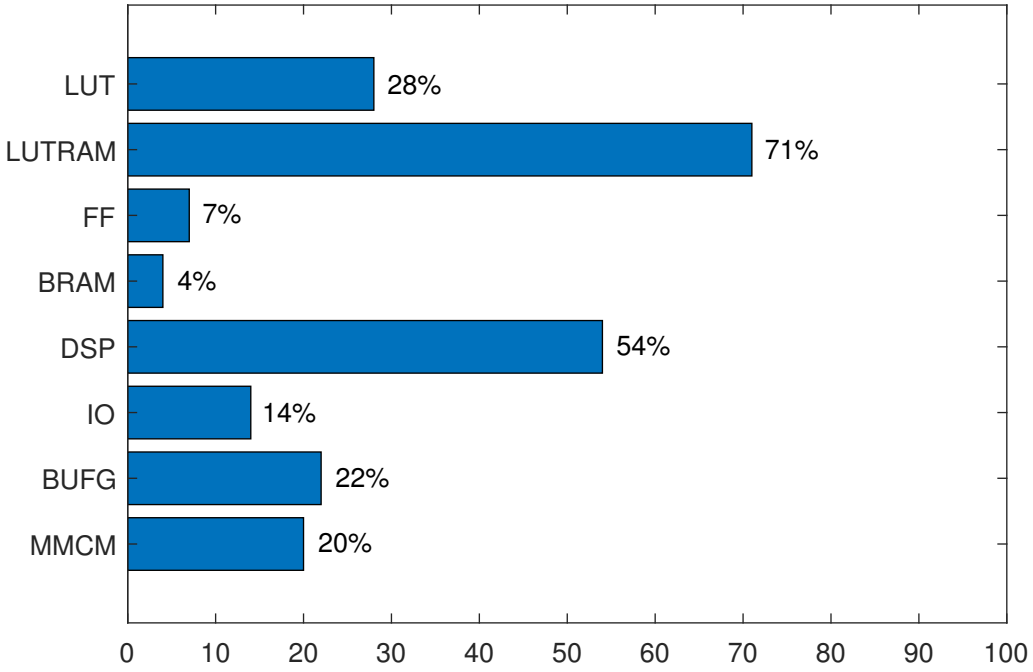


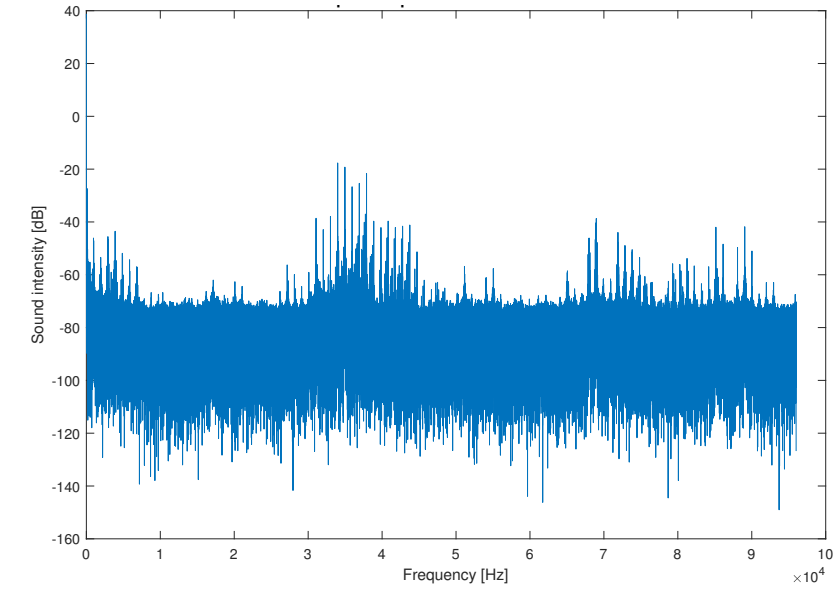
Figure 4.3: This figure displays the FPGA resource utilization of the system using 48 sensors.

In Fig. 4.4a the frequency spectrum of the ultrasonic transducer collected using the `Static Beam` mode. This is the only sound source we have had access to which reliably emits sound of which the majority resides in the ultrasonic range. A lower frequency sound generated by a tone generator from a cell phone is shown in Fig. 4.4b. This result is what facilitated more accurate simulation results in Section 4.1.

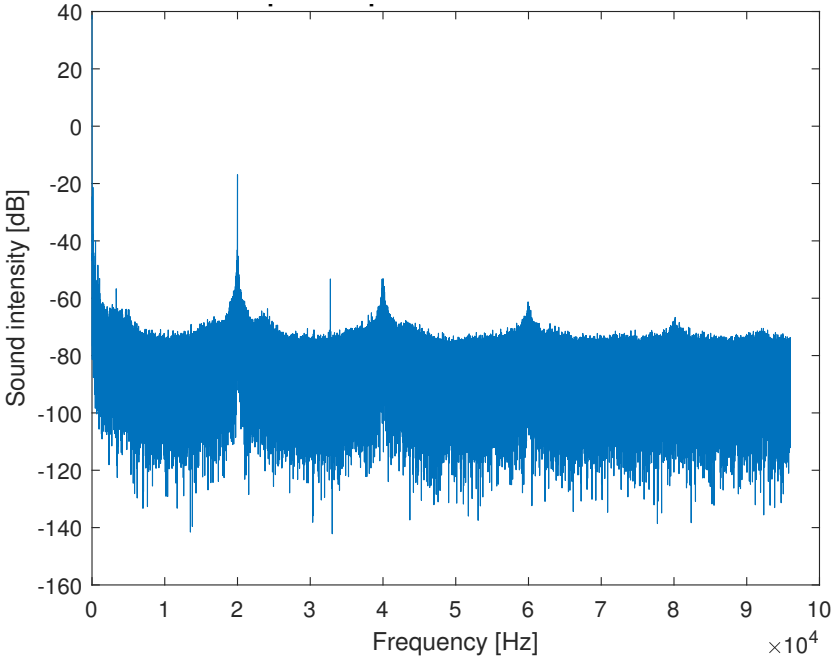
In Fig. 4.6 the results from testing the system with 24 sensors with the 35 kHz sound source can be seen. The patterns are very similar to those of Fig. 4.1, which tells us the DAS algorithm has been implemented correctly. Note that the beamform pattern is now limited to $30\,^{\circ} - 150\,^{\circ}$, the reason for which is explained in Section 3.3.

Upon multiplexing the whole system and separating it in MATLAB as described in Section 3.2.7 two directional energy vectors are attained which can be seen plotted in Fig. 4.7. It was at this point possible to verify in real time that the θ plot responded to horizontal movements and the ϕ plot responded to vertical movement and work on attaining the outer product of the two vectors could begin.

The results of the first test with a heatmap can be seen in Fig. 4.8. It is apparent



(a)



(b)

Figure 4.4: This figure displays the analysis result of the spectrum of the two sources. Subfigure (a) shows the ultrasonic source at $f = 35\,\text{kHz}$ and (b) shows a sine wave at $f = 20\,\text{kHz}$ generated by a cell phone.

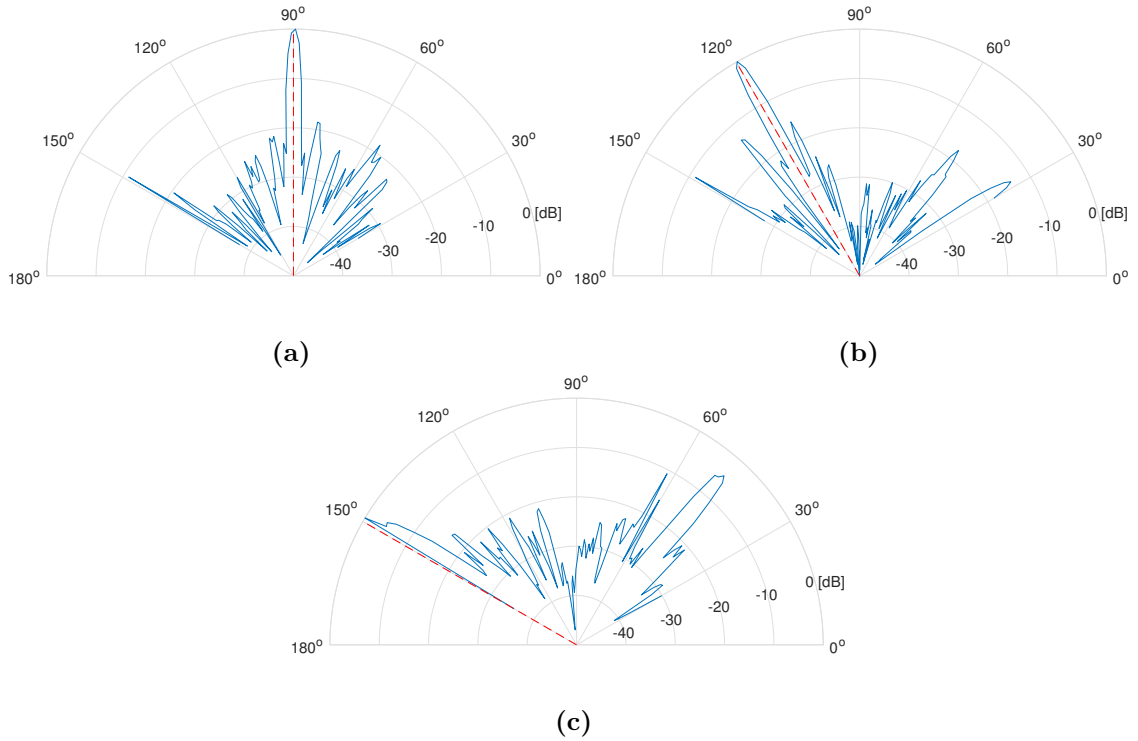


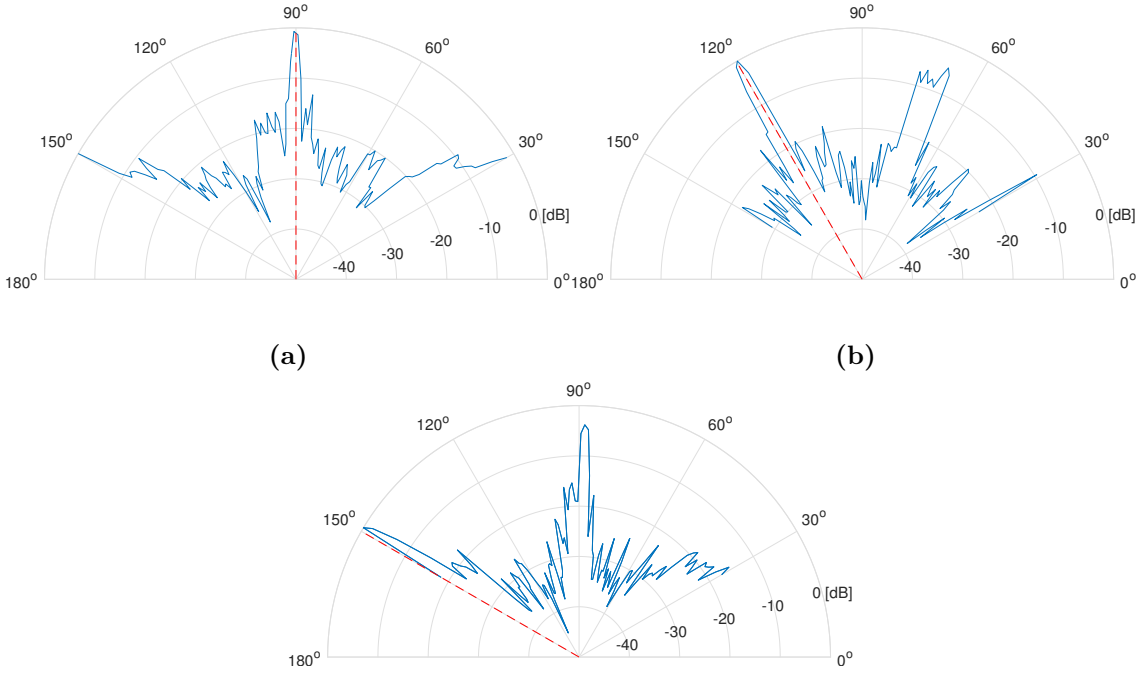
Figure 4.5: This figure displays the test result of the implemented system with 24 sensors spaced $d = 11.43\,\text{mm}$ apart using a sound source with $f = 20\,\text{kHz}$ placed at (a) $\theta = 90\,^{\text{o}}$, (b) $\theta = 120\,^{\text{o}}$ and (c) $\theta = 150\,^{\text{o}}$.

that the source position is more clearly discernible in a linear scale which is why all subsequent test results will be presented this way.

Testing with the ultrasonic source provided an ambiguous pattern in the heatmap, as can be seen in Fig. 4.9, but the highest directional energy appears in the correct place and both angles displayed are correct as well.

The purpose of the next test is to find out how the distance between the source and the arrays affects the system's performance. It was carried out by placing a sound source of 20 kHz and various distances right in front of the arrays. The results can be seen in Fig. 4.10 where it becomes apparent that the system's ability to locate the source deteriorates drastically somewhere between 0.5 m - 1.0 m. The strongest candidate appears at $\theta = 90\,^{\text{o}}$ and $\phi = 30\,^{\text{o}}$. Additionally at 4.0 m there appears to be mirror images near $\theta = 90\,^{\text{o}}$ and $\phi = 90\,^{\text{o}}$. In general, mirror images seem occasionally to appear in the vertical directional energy vector, but never in the horizontal directional energy vector.

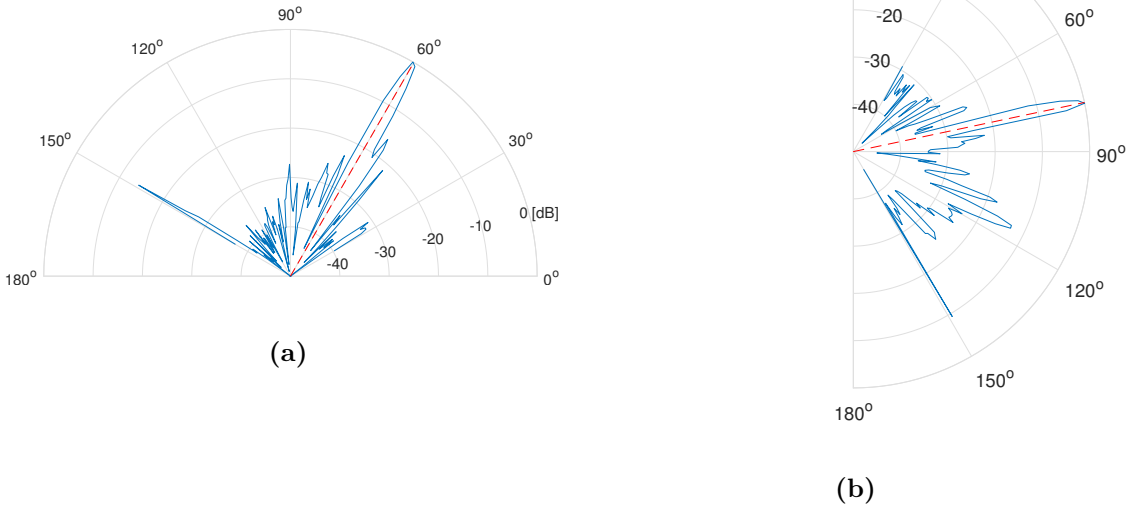
A great deal of testing with transient sounds such as claps, finger snaps and rattling with keys have been tried with the array. All such testing has strongly indicated that the system is incapable of locating such sounds, i.e. sounds that last for fractions of a second. Ostensibly, sounds had to last for roughly one second for there to be a distinguished peak of energy in the heatmap.



(a) (b)

(c)

Figure 4.6: This figure displays the test result of the implemented system with 24 sensors spaced $d = 11.43\,\text{mm}$ apart using a sound source with $f = 35\,\text{kHz}$ placed at (a) $\theta = 90\,°$, (b) $\theta = 120\,°$ and (c) $\theta = 150\,°$.



(a)

(b)

Figure 4.7: This figure displays the results of the implementation of two perpendicular arrays with 48 sensors by plotting θ and ϕ separately, (a) shows θ and (b) shows ϕ. The sound source is of the characteristics shown in Fig. 4.4b and is arbitrarily placed in the room.

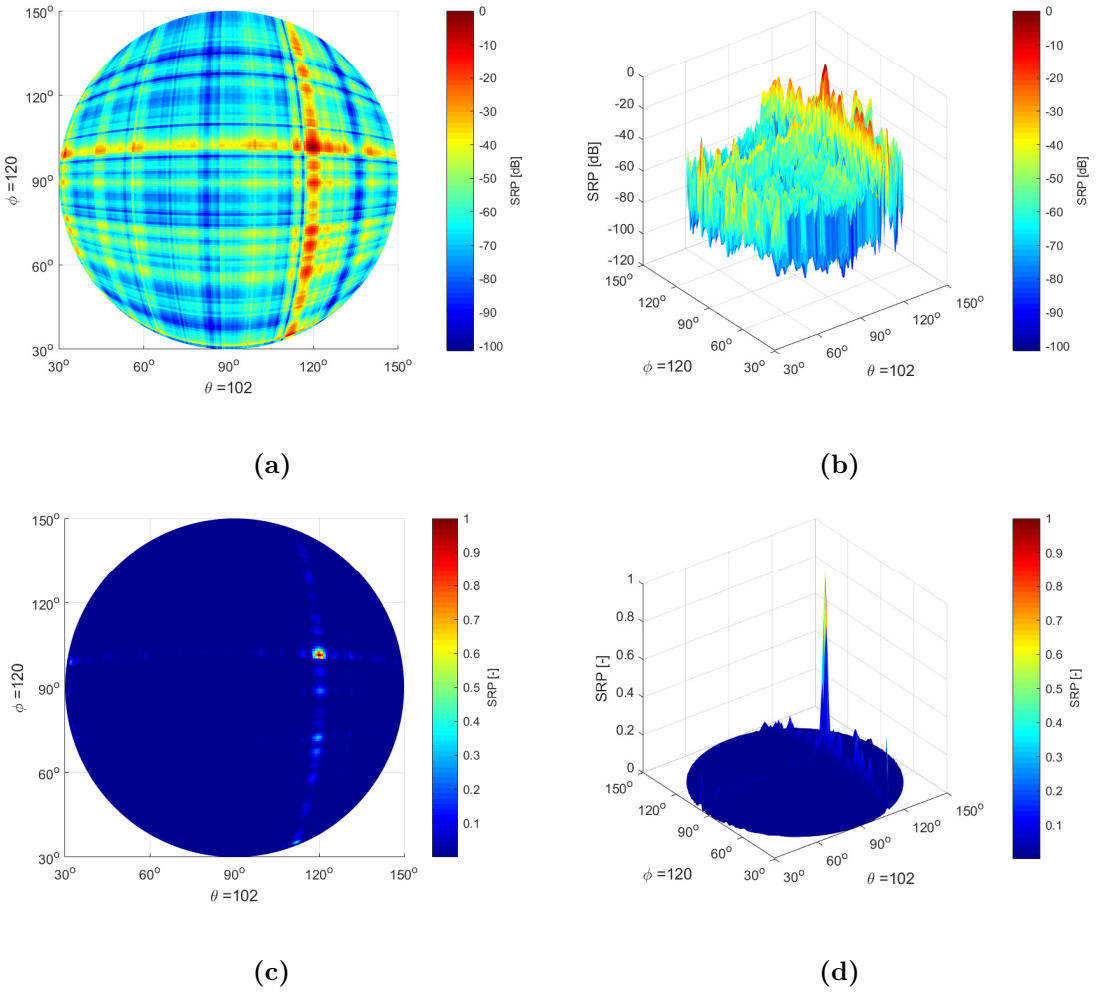
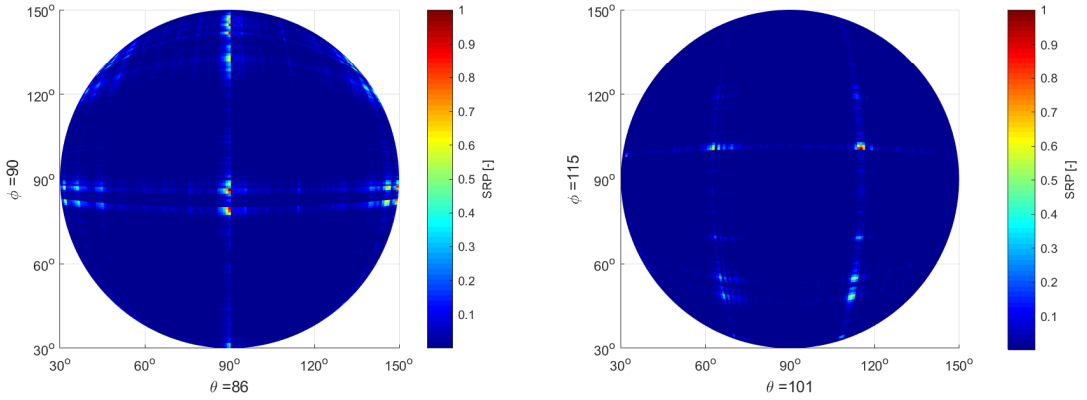


Figure 4.8: This plot shows the result of taking the outer product of the two directional energy vectors. Subfigures (a) and (b) show the plot on a logarithmic scale and (c) and (d) show the plot on a linear scale.



(a) **(b)**

Figure 4.9: This image shows the results of testing source localization with the ultrasonic transducer which is of the characteristics shown in Fig. 4.4a. The sound source were placed at (a) $\theta = 90\,^{\circ}$ $\phi = 90\,^{\circ}$ and (b) $\theta = 102\,^{\circ}$ $\phi = 90\,^{\circ}$.

Testing with steady sources in noisy environments showed that the sound source is easily drowned out in sounds such as a normal conversation. Even sounds from our footsteps as we performed the test could occasionally have a noticeable effect on the system. Especially sounds at low frequency, specifically any sound below 4 kHz would notably result in chaotic patterns in the heatmap. It should be noted that since our system gathers 50 samples per angle at 192 kHz it only captures audio for $50 \cdot (192\,\text{kHz})^{-1} \approx 260\,\mu\text{s}$ and can thus only capture a full wavelength, λ, of the sound if $\lambda \leq c \cdot 260\,\mu\text{s} = 89.18\,\text{mm}$, where $c = 343\,\text{m/s}$. This means that if the frequency is below $c/\lambda \approx 3846\,\text{Hz}$ the system will capture less than one wavelength of the sound.

Finally, as mentioned in Section 3.4 tests were carried out to estimate the system's sensitivity to small movements of the sound source, i.e. to estimate the system's accuracy. The most interesting question to answer was the one raised in Section 3.3 regarding whether the system is actually able to distinguish between all the 128 discrete angles. Because the problems with spatial aliasing persisted and were more prevalent with the ultrasonic source these accuracy tests were performed with the lower-frequency source of 20 kHz. The testing method was to simply place the source in front of the array, note the angles calculated by MATLABs `max` function and then move the source in the horizontal direction until the displayed value for θ changed. Unfortunately, the results of this test must be said to be inconclusive because the calculated angles generally tended to fluctuate between two values.

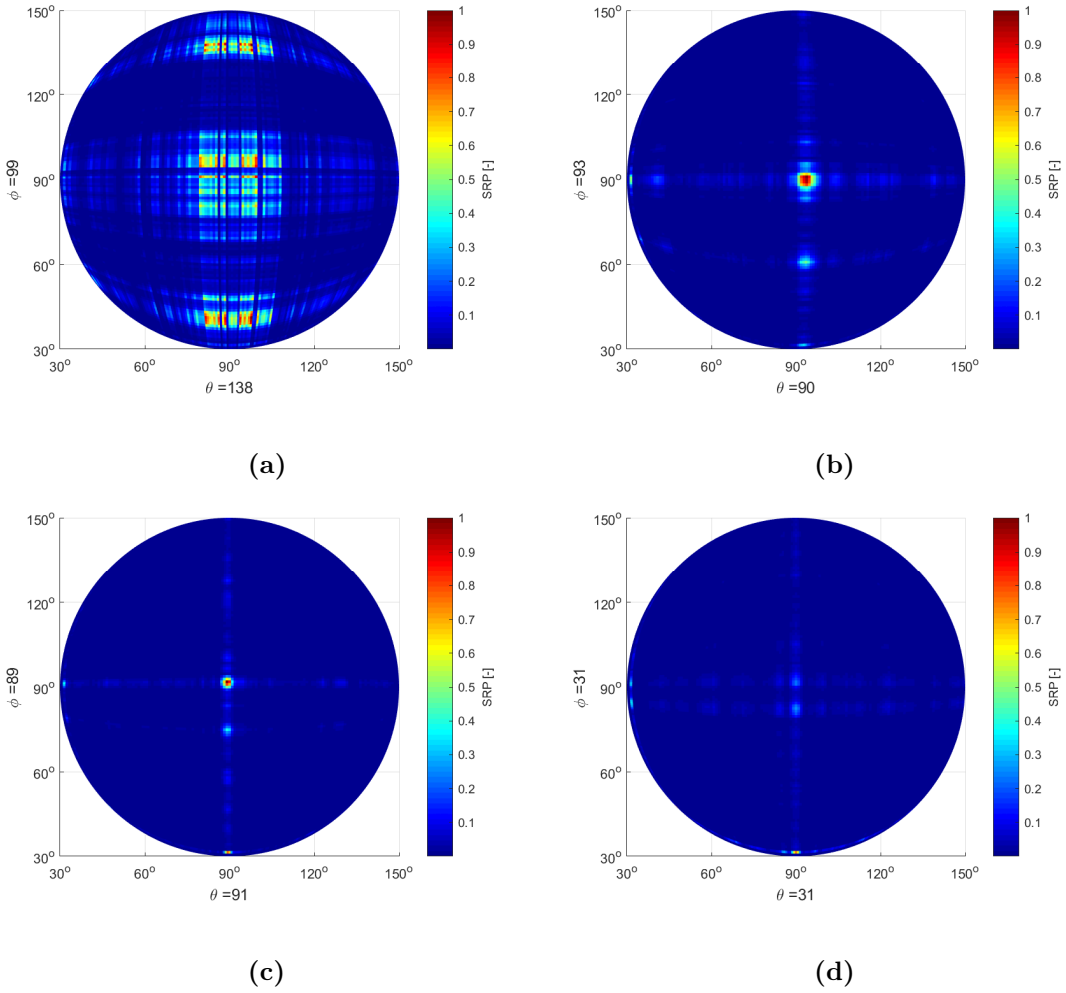


Figure 4.10: This figure shows the results of testing the system with a sound source at different distances from the array which is of the characteristics shown in Fig. 4.4b and at $\theta = 90\,°$ and $\phi = 90\,°$. The distance is (a) 0.5 m, (b) 1.0 m, (c) 2.0 m and (d) 4.0 m.

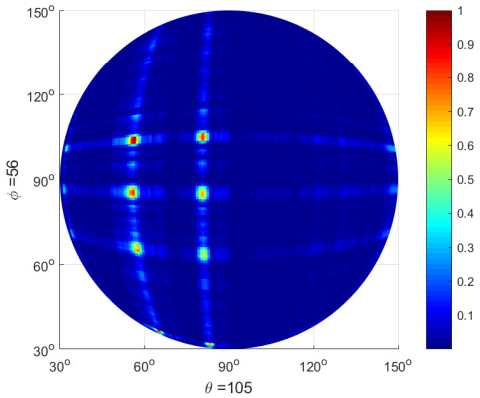


Figure 4.11: Two sound sources at 20 kHz.

5

Discussion and Possible Improvements

The chapter analyzes the results and observations presented in Chapter 4. First the reasons for the results are discussed and this is followed by suggestions on possible improvements that can be made to the system.

5.1 Discussion

A system has been designed that employs two one-dimensional arrays to create a heatmap representing the directional energy in two dimensions.

5.1.1 General performance

As is clear from the results it is possible to perform localization on the source we are using in our experiments, but this localization is not very reliable. According to the simulation results shown in Fig. 2.4 spatial aliasing should become less of a problem when power is more evenly distributed over a broad range of frequency, which we were partly successful in verifying. The ultrasonic sound source we had access to emitted sound in a rather narrow band so the difference in amplitude between the main lobe and the grating lobes was just barely distinguishable. Sources for ultrasound that span over broader bands of frequency must be employed to verify the system's capabilities, but it is far from trivial to produce such a source as any equipment that produces sound at ultrasonic frequencies is not common. Research into how to produce such a source should be carried out. It would also be interesting to use ultrasound that has the spectral characteristics of animals that vocalize in the ultrasonic range.

5.1.2 Brief sounds

Currently, the system cannot pick up sounds that last a very brief time, such as clapping and snapping sounds. A possible reason for this is the low system frequency, f_{sys}. As mentioned in Section 3.2.4 it takes 96 ms for one full sweep to be finished. This means that a sound which lasts a shorter time than this could occur when the beamformer is pointing in another direction and a few milliseconds later, when the beamformer is pointing in the direction which the sound originated from, the sound could be gone. This would mean that the sound at its highest intensity would not be picked up by the main lobe of the beamformer, but would nevertheless be strong enough to overpower all other directional energy values. This would then result in a strong directional energy in the wrong location on the heatmap.

One potential way for eliminating this problem would be to completely revise the system so that it instead works by storing one audio sample set in a buffer for each sensor and performing all processing on the same sets of sampled audio. This idea is illustrated in Figs. 5.1a and 5.1b where four sampled sets are being cropped out at different times. In Figs. 5.1c and 5.1d the resulting signals from adding the cropped signals show that there is a combination of time-shifted data that results in a maximum correlation which suggests that this method could be equivalent to the DAS algorithm, if not mathematically then at least practically. The implications of this would be that, while the audio data would have to be stored at 4.608 MHz, the algorithm could be carried out at the maximum tolerable rate of the FPGA - 450 MHz [26]. Theoretically, this also means that all combinations of time-shifted data, i.e. data from all angles, could be processed in parallel which would drastically reduce latency. In the current implementation there was a lot of focus on the RAM-based shift register IP which would not be used in such an implementation as just described. The approach of storing audio would require a RAM that allows for accessing any and all portions of the stored audio; this approach would essentially require redesigning the entire system.

5.1.3 FPGA Resource Utilization

As seen in Fig. 4.3 the the LUTRAM resources come closest to being exhausted. The resources that are a secondary candidate of being exhausted are the DSP slices.

The high consumption of LUTRAM resources is a result of the way the delay lines are implemented; there is a need for large delays to allow for large arrays and there is a need for small delays to allow for finer scanning resolution. These two requirements combined results in a prohibitively large consumption of LUTRAMs. The most straightforward remedy to this would be to just use one shift register IP. This is a highly recommended course of action as the solution of using three cascaded shift register IPs did not work entirely as intended; the main incentive was to have a smaller minimum delay while still being able to produce the required maximum delay (which is derived from the size of the physical size of the array),

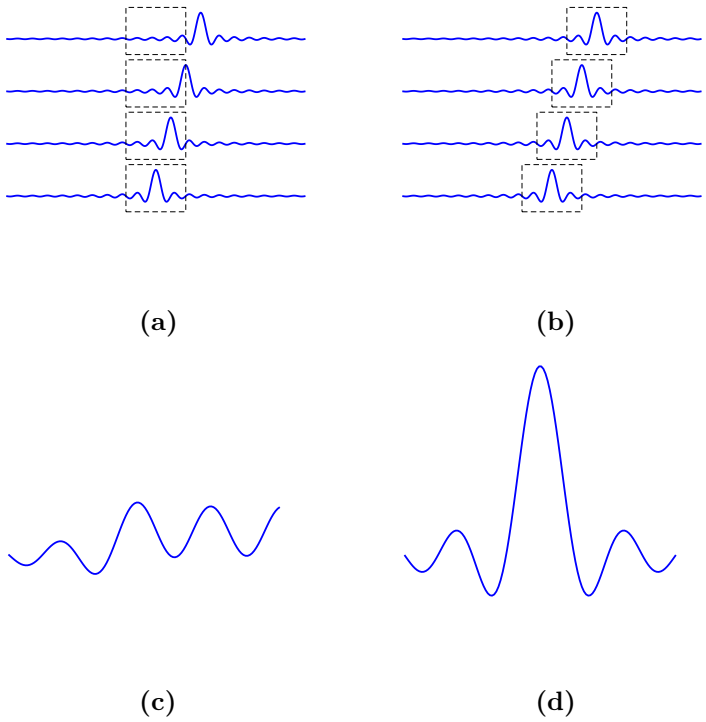


Figure 5.1: This figure illustrates the idea of performing all processing on the same sets of audio data using an example with four audio streams. The blue waveforms in (a) and (b) represent sampled audio coming from some direction (and thus arriving at the sensors at different times) and the dashed boxes represent the act of selecting a portion of the data for further processing. In (a) the four audio streams are cropped with no time shift and if the data from these windows are added together it results in the waveform seen in (c). In (b) the four audio streams are cropped with a time shift causing the waveforms to synchronize and if the data from these windows are added together it results in the waveform seen in (d).

but the minimum delay possible with this solution turned out to be one clock cycle per shift register IP, i.e. three clock cycles. Currently the only benefit from this solution is that the need for clock-domain crossing was eliminated.

Another, more ambitious, approach for reducing the consumption of LUTRAMs would be to take the approach of storing audio in buffers and perform all processing on these same sets of audio samples, as described in Section 5.1.2. This would eliminate the need for the shift register IPs altogether and instead require, if there is sufficient space for it, only one block RAM for each audio stream. The FPGA used in this project has 445 available block RAM blocks of 36 kbit [34] which at 192 kHz facilitates $(192\,\text{kHz} \cdot 16b)^{-1} \cdot 36\,\text{kB} \approx 11.72\,\text{ms}$ of stored audio in each block RAM. Sound travelling at 343 m/s will have travelled about four metres in 11.72 ms, which would suggest that a block RAM of 36 kb is more than sufficient to store all the data needed for producing the delays required for a very large array.

To bring down the high consumption of DSP slices the multichannel capabilities of the CIC filter and FIR filter IPs should be exploited. As seen in Fig. 5.2 several data streams would be multiplexed and run through the filters. It is possible to run the filters at a much higher clock rate than 4.608 MHz, which means that one single chain of filters could handle several streams simultaneously without having to slow down the rate of the data.

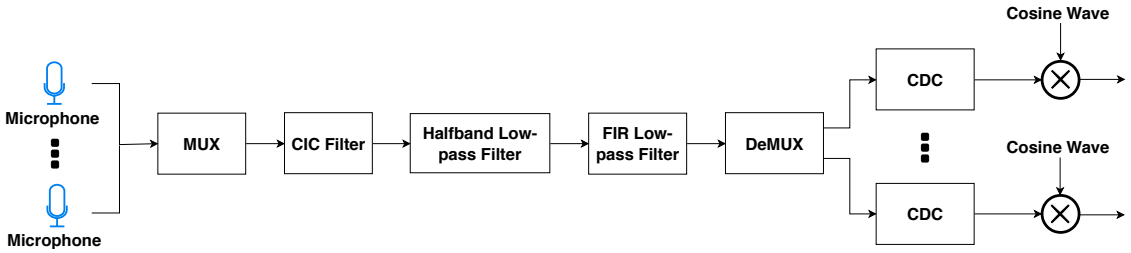


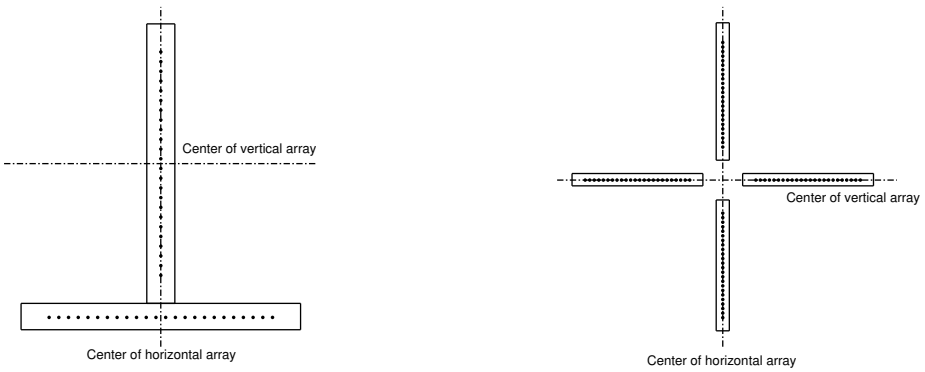
Figure 5.2: This image illustrates how the chain of filters that perform the PDM-to-PCM conversion could be serialized.

5.1.4 Mirror images in the vertical array

It can be seen from some of the results that there is often an mirror image present in the vertical directional energy component of the heatmap. This is especially obvious in Fig. 4.9a and Fig. 4.10d. A mirror image in the horizontal, however, has never been observed during testing. A possible reason for this could be that, as can be seen in Fig. 5.3a, the center of the horizontal array is aligned with the vertical board, but the center of the vertical array is offset from the horizontal board. If this is the case then the problem should disappear when the system is expanded to use four sensor boards, because at this point an entirely symmetrical configuration, as the one suggested in Fig. 5.3b, could be constructed.

5.1.5 Noisy environment

It is very clear from testing that ambient sound, such as people conversing, adversely inhibits the systems ability to distinguish sound sources. A great improvement to the system could be to combat such effects by high-pass filtering the sound. This filtering stage would have to be put somewhere before the integration stage in the FPGA; it is not possible to do it on the PC end.



(a) **(b)**

Figure 5.3: This figure displays (a) the geometrical configuration of the current array and (b) a symmetrical configuration for a system using four sensor boards.

5.2 Potential improvements

There are some potential improvements that could be made to the system. This section contains a list of recommended actions arranged in descending order of urgency.

Investigating how to pick up short-duration sounds

The systems inability to locate brief sounds is perhaps the most unfavorable problem with the current system; many applications may require the array to be capable of picking up sounds of very short duration, such as footsteps. At the moment the only hypothesis we have as to why the system cannot locate brief sounds relates to the latency of the system, so our recommendation would be to investigate this.

Implementing a high-pass filter

As mentioned in Section 5.1.5 the system does not perform well in noisy environment. Arguably the most important improvements is to implement a high-pass filter to remove low frequency components, which is likely to drastically improve the system's performance. For instance, FIR high-pass filters could be implemented for each data

stream as seen in Fig. 5.4a and thus consume 24 DSP slices since one FIR high-pass filter consumes one DSP slice which can be known from FIR IP in Vivado. It is also conceivable that a single high-pass filter could be implemented after the DAS algorithm as seen in Fig. 5.4b and thus consume only a single DSP slice.

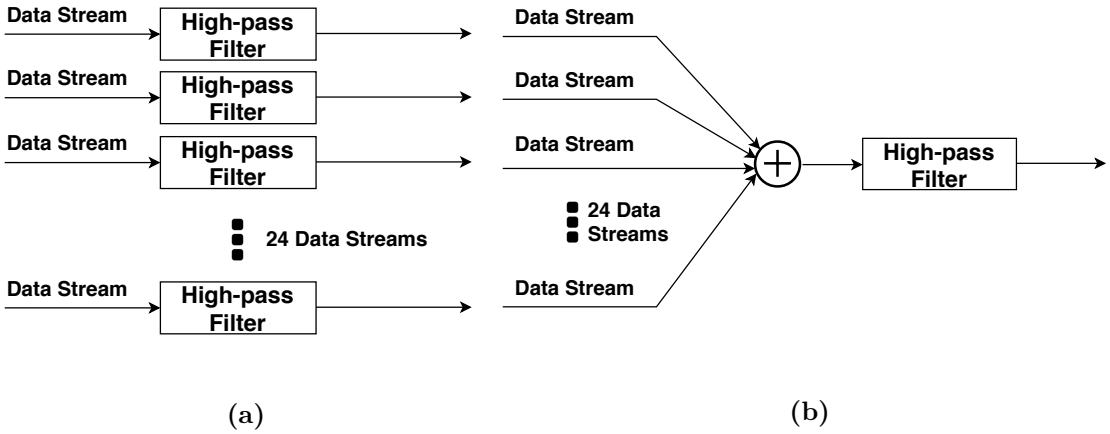


Figure 5.4: This figure displays (a) implementing high-pass filter in parallel and (b) implementing high-pass filter in serial.

Investigating ways to circumvent the fundamental delay

Some literature suggest systems that use much larger arrays compared to the one used in this project. This would mean the time for the sound to propagate from the first microphone to the last would be too large to be able to achieve real-time localization with the approach we are using. This indicates to us that there are methods to improve the system in terms of latency and this would have to be investigated.

Refining the transfer interface between FPGA and PC

Currently the transfer of information between the host computer and the FPGA is one-sided; information can only be sent from the FPGA to the host computer. As explained in Section 3.2.5 the angle index and `Board Flag` is sent along with the directional energy data, which is a non-ideal solution. A natural refinement to the system would be to allow for the host computer to send a signal to the FSM so that it is known which set of directional energy data will be transferred first.

Serializing the PDM to PCM conversion

One way to save FPGA resources is possible to serialize the PDM to PCM conversion. The filters could potentially be run at a much higher frequency and several audio data streams could be multiplexed to save hardware resources (see Fig. 5.2).

Using ring buffers

To avoid unnecessary storage of calculated delay values the possibility of using ring buffers should be investigated.

6

Conclusion

The goal of this master thesis was to design a system that uses a massive MEMS microphone array and beamforming to perform real-time positioning of ultrasonic sources. Another key objective was to maximize the amount of microphones in the array, since this improves the accuracy of which the array can localize sound sources, and using an FPGA was hypothesized to be a suitable tool for this purpose.

A system that can perform sound source localization has been designed and it can, arguably, be said to operate in real time at a rate of 5.2 locations per second. The localization is visualized in a heatmap which displays energy originating from different directions.

In regards to the goal of maximizing the number of microphones: the final version of the system uses 48 microphones. The reason it was not feasible to increase this number was because of the system's large latency, which originates from the way the system implements the DAS algorithm, the speed of sound and the physical size of the array. If the latency were to become even higher the system could no longer be said to operate in real time. Thus it was this latency, and not a shortage of FPGA resources, that limited the number of microphones, leading to the conclusion that the first step of any future development should be to completely revise how the system implements the DAS algorithm as discussed in Section 5.1.2. If this revised version of the system could be implemented and the new limiting factor instead becomes high consumption of FPGA resources there are ways to reduce the resource consumption as discussed in Section 5.1.3.

Any conclusions regarding the accuracy of the system were, as stated in Section 4.2, difficult to make. What can be said for certain is that the fundamentally smallest angle difference the system can distinguish is 0.9375° but it is possible that this angle is slightly larger in practice, as discussed in Section 3.3. Decreasing this angle, and thus increasing the resolution of the heatmap, would lead to a larger latency. Again, it must be concluded that the way the system implements the DAS algorithm should be revised.

Bibliography

[1] L. Dai, *Intelligent Macromolecules for Smart Devices*. Springer London, Jan. 2004, pp. 451–452.

[2] H. Krim and M. Viberg, "Two decades of array signal processing research: The parametric approach", *IEEE Signal Processing Magazine*, vol. 13, no. 4, pp. 67–94, Jul. 1996.

[3] S. Li and M. Stanaćević, "Source separation in noisy and reverberant environment using miniature microphone array", in *2014 48th Asilomar Conference on Signals, Systems and Computers*, Nov. 2014, pp. 446–449.

[4] A. R. Abu-El-Quran, R. A. Goubran, and A. D. C. Chan, "Security monitoring using microphone arrays and audio classification", *IEEE Transactions on Instrumentation and Measurement*, vol. 55, no. 4, pp. 1025–1032, Aug. 2006.

[5] D. T. Blumstein, D. J. Mennill, *et al.*, "Acoustic monitoring in terrestrial environments using microphone arrays: Applications, technological considerations and prospectus", *Journal of Applied Ecology*, vol. 48, no. 3, pp. 758–767, Jun. 2011.

[6] J. Steckel and H. Peremans, "Ultrasound-based air leak detection using a random microphone array and sparse representations", in *IEEE SENSORS 2014 Proceedings*, Nov. 2014, pp. 1026–1029.

[7] E. E. Case, A. M. Zelnio, and B. D. Rigling, "Low-cost acoustic array for small UAV detection and tracking", in *2008 IEEE National Aerospace and Electronics Conference*, Jul. 2008, pp. 110–113.

[8] J. Benesty, J. Chen, and Y. Huang, *Microphone Array Signal Processing*. Springer-Verlag Berlin Heidelberg, 2008, pp. 40–46, 192–193.

[9] M. Fallahi, M. Blau, *et al.*, "Optimizing the microphone array size for a virtual artificial head", *The international Symposium on Auditory and Audiological Research (ISAAR)*, vol. 6, pp. 359–366, Jan. 2018.

[10] M. Brandstein and D. Ward, *Signal Processing Techniques and Applications*. Springer-Verlag Berlin Heidelberg, 2001, pp. 158–164, 181–215.

[11] R. Etienne-Cummings and M. Clapp, "Architecture for source localization with a linear ultrasonic array", in *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No.01CH37196)*, vol. 3, May 2001, 181–184 vol. 2.

[12] J. Grythe, *Beamforming algorithms - beamformers*, `https://wpstatic.idium.no/web2.norsonic.com/2016/10/TN-beamformers.pdf`, Oslo, Norway.

[13] J. Tiete, F. Domínguez, *et al.*, "Soundcompass: A distributed MEMS microphone array-based sensor for sound source localization", *Sensors (Basel, Switzerland)*, vol. 14, no. 2, pp. 1918–1949, Jan. 2014.

[14] D. Petersen and C. Howard, "Simulation and design of a microphone array for beamforming on a moving acoustic source", *Acoustics 2013 - Victor Harbor*, Nov. 2013.

[15] U. Farooq, Z. Marrakchi, and H. Mehrez, *Tree-based Heterogeneous FPGA Architectures, Application Specific Exploration and Optimization*. Springer-Verlag New York, 2012, pp. 7–8.

[16] B. Goel, *Introduction to FPGA*, 2017.

[17] D. Todorović, I. Saln, *et al.*, "Implementation and application of FPGA platform with digital MEMS microphone array", in *4th International Conference on Electrical, Electronics and Computing Engineering*, Kladovo, Serbia, Jun. 2017, AKI2.2.1–6.

[18] Xilinx. (May 2018). Company Overview, [Online]. Available: `https://www.xilinx.com/about/company-overview.html` (visited on 06/19/2018).

[19] ——, (May 2018). Vivado Design Suite - HLx Editions, [Online]. Available: `https://www.xilinx.com/products/design-tools/vivado.html` (visited on 06/19/2018).

[20] D. S. Jacobs and A. Bastian, *Predator–Prey Interactions: Co-evolution between Bats and Their Prey*. Springer International Publishing, 2016.

[21] R. Mucci, "A comparison of efficient beamforming algorithms", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 3, pp. 548–558, Jun. 1984.

[22] B. Zimmermann and C. Studer, "FPGA-based real-time acoustic camera prototype", in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, May 2010, pp. 1419–1419.

[23] D. E. Dudgeon, "Fundamentals of digital array processing", *Proceedings of the IEEE*, vol. 65, no. 6, pp. 898–904, Jun. 1977.

[24] J. Dmochowski, J. Benesty, and S. Affes, "On spatial aliasing in microphone arrays", *IEEE Transactions on Signal Processing*, vol. 57, no. 4, pp. 1383–1395, Apr. 2009.

[25] *Digital zero-height sisonic microphone with multi-mode and ultrasonic support*, Knowles Electronics, May 2014.

[26] *Genesys 2 FPGA board reference manual*, Digilent, Aug. 2017.

[27] *Ultrasonic transducer*, Kobitone Audio Company, May 2006.

[28] R. Lyons, "Understanding cascaded integrator-comb filters", Besser Associates, Tech. Rep., Mar. 2005.

[29] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals & systems*, English, 2. ed. Upper Saddle River, N.J: Prentice Hall, 1997.

[30] *RAM-based shift register v12.0*, XILINX, Nov. 2015.

[31] *Digilent adept parallel transfer interface (DPTI) programmer's reference manual*, DIGILENT, Jun. 2015.

[32] *Digilent parallel transfer interface (DPTI)*, DIGILENT, Jun. 2016.

[33] R. Ginosar, "Metastability and synchronizers: A tutorial", *IEEE Design Test of Computers*, vol. 28, no. 5, pp. 23–35, Sep. 2011.

[34] *7 series FPGAs memory resources*, XILINX, Sep. 2016.