Multi-channel microphone system for sound localization and beamforming

Davíð Freyr Jónsson

Lokaverkefni í rafmagnstæknifræði BSc

2016

Höfundur: Davíð Freyr Jónsson
Kennitala: 2001824569
Leiðbeinandi: Baldur Þorgilsson

Tækni- og verkfræðideild
# Tækni- og verkræðideild

## Heiti verkefnis:
Multi-channel microphone system for sound localization and beamforming

<table>
<thead>
<tr>
<th>Námsbraut:</th>
<th>Tegund verkefnis:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rafmagnstæknifraði BSc</td>
<td>Lokaverkefni í tæknifraði BSc</td>
</tr>
</tbody>
</table>

## Önn:
Vor 2016

## Námskeið:
RT
LOK1012

## Árip:
This project describes the design of multi-channel microphone system for research in the field of sound localization and beamforming. The system should be capable of sampling data from up to 128 microphones simultaneously and stream in real time to a PC. The system consists of 16 analog to digital module with 8 microphone channels each. Audio data is to be multiplexed on one USB cable. The aim is to create a system that is flexible and easy to use.

## Höfundur:
Davíð Freyr Jónsson

## Umsjónarkennari:
Baldur Þorgilsson

## Leiðbeinandi:
Baldur Þorgilsson

## Fyrirtæki/stofnun:
Háskólinn í Reykjavík

## Dagsetning:
6. maí 2016

## Lykilrð íslensk:
<table>
<thead>
<tr>
<th>Hljóð</th>
<th>Hljóðnemi</th>
<th>Hljóðnemaröð</th>
</tr>
</thead>
</table>

## Lykilrð ensk:
| Sound | Microphone array | Sound localization |

## Dreifing:
opin ❌ lokuð ❌ til:
Abstract

This project describes the design of multi-channel microphone system for research in the field of sound localization and beamforming. The system should be capable of sampling data from up to 128 microphones simultaneously and stream in real time to a PC. The system consists of 16 analog to digital module with 8 microphone channels each. Audio data is to be multiplexed on one USB cable. The aim is to create a system that is flexible and easy to use.

Reykjavík, 4. May 2016

____________________________

Davíð Freyr Jónsson
Contents

Abstract I
List of figures V
List of tables VI
Introduction 1
Abbreviations 4

1 Multi-channel microphone system 5
  1.1 System requirements .............................. 5
  1.2 Goals for initial design ........................... 6
  1.3 Design topologies ................................. 7
      1.3.1 Design topology 1 ............................ 7
      1.3.2 Design topology 2 ............................ 8
      1.3.3 Design topology 3 ............................ 9
      1.3.4 Choosing a design topology .................. 10

2 ADC module design 11
  2.1 Analog to Digital Converter ...................... 11
      2.1.1 Left-Justified serial audio format ............. 13
      2.1.2 I²S serial audio format ....................... 14
      2.1.3 TDM serial audio format ....................... 14
      2.1.4 Choosing the serial audio format ............... 15
      2.1.5 CS5368 pin and connection diagram .............. 15
      2.1.6 CS5368 clock sources and sampling synchronization .... 16
      2.1.7 CS5368 clock frequency ........................ 17
      2.1.8 CS5368 reference voltage ...................... 17
      2.1.9 CS5368 DC power consumption ................... 18
  2.2 ADC input drivers ................................. 18
  2.3 MCU for ADC module ................................ 22
  2.4 Voltage regulators ................................. 24
      2.4.1 Thermal resistance and input voltage .......... 26
  2.5 PCB layout and noise filtering ..................... 29
      2.5.1 Power rail noise filtering ..................... 30
      2.5.2 PCB layout and noise .......................... 32
3 Streaming audio data to PC
  3.1 System data rate .................................................. 36
  3.2 USB 3.0 module ....................................................... 36
  3.3 FPGA: Interface the EZ-USB FX3 .................................. 38
    3.3.1 FPGA programming ............................................. 41

4 Results ................................................................. 43
  4.1 Building and testing the ADC module ............................. 43
    4.1.1 Design flaws in ADC module ................................... 44
    4.1.2 Noise measurements: Power rails ............................ 46
    4.1.3 Noise measurements: Input drivers .......................... 49
    4.1.4 Heat dissipation .............................................. 50
  4.2 Testing the FPGA module ......................................... 50
    4.2.1 MachXO3L problems ........................................... 52
  4.3 USB module: EZ-USB FX3 .......................................... 52

5 Conclusion ............................................................ 53
  5.1 Future work ......................................................... 54
    5.1.1 ADC module ..................................................... 54
    5.1.2 FPGA module ................................................... 54
    5.1.3 EZ-USB FX3 module ............................................ 55

References .............................................................. 55

Appendix ................................................................. 60

A ADC module ........................................................... 60
  A.1 Schematic diagrams ............................................... 60
  A.2 PCB layout ......................................................... 67

B FPGA ................................................................. 72
  B.1 VHDL test program for TDM SIPO/FIFO module .................. 72
List of Figures

1. The Eigenmike microphone from mh acoustics consists of a total of 32 microphones arranged in a sphere. [2] ........................................ 1
2. Microphone array consisting of 1020 microphones, a project at MIT. [3] .... 2
1.1 Design topology 1. ........................................ 8
1.2 Design topology 2. ........................................ 9
1.3 Design topology 3. ........................................ 10
2.1 Block diagram of the CS5368 ADC. [8] ........................... 13
2.2 Left-justified audio format. [8] ............................... 14
2.3 I2S audio format. [8] ....................................... 14
2.4 TDM audio format. [8] ....................................... 14
2.5 Pin and connection diagram of the CS5368 ADC. [8] ........................... 16
2.6 Synchronizing the sampling of many CS5368. [8] ........................... 17
2.7 Table from CS5368 datasheet showing DC power consumption. [8] .... 18
2.8 Input driver for the CS5368 ADC as suggested by Cirrus Logic in datasheet. [8] 19
2.9 An example of ADC driver with differential outputs using THS4521 operational amplifier form Texas Instrument. [10] ........................................ 19
2.10 The input driver for the ADC module using the THS4521 operational amplifier. 22
2.11 The program for ATtiny167 MCU on ADC module .......................... 24
2.12 SOT223 package used for the LP38693 voltage regulator. [14] .................. 26
2.13 Placement of the LP38693 voltage regulators, U5 and U6, on top layer. ........ 28
2.14 Bottom layer showing the vias which connect the ground plains both electrically and thermally. 28
2.15 Top layer of the PCB for ADC module. .................................. 30
2.16 Bottom layer of the PCB for ADC module. .................................. 30
2.17 Filter for the 5 V analog power rail. .................................... 31
2.18 Frequency response of the TDK 0.01µF capacitor. [18] ...................... 31
2.19 Frequency response of the TDK 1µF capacitor. [19] .......................... 32
2.20 The return current distribution under a signal trace. The current stays under the trace. [21] ........................................ 33
2.21 Preferred ground plane layout. Analog and digital ground planes are connected at one point and all signal traces between the two parts are routed through that point. [21] ........................................ 33
2.22 The PCB for the ADC module has separate ground for the digital part and analog part, only connecting at a common point (red circle). No return currents from components producing high-frequency signals should have to pass through the analog part. ........................................ 34
3.1 Cypress EZ-USB FX3 USB 3.0 module. [26] ............................. 37
3.2 GPIF Designer from Cypress is a powerful tool to configure the EZ-USB FX3 module. [29] 
3.3 FPGA architecture. [30] 
3.4 A block diagram which shows the function of the SIPO/FIFO module implemented in an FPGA. 
3.5 Lattice MachXO3L Starter Kit. [33] 
3.6 A CLB as shown in Lattice Diamond programming environment for the LCMXO3L-6900C. 
3.7 A physical overview form Lattice Diamond of the interconnections between CLB after a VHDL code has been synthesized and routed. 

4.1 Bottom layer showing the vias which connect the ground plains both electrically and thermally. 
4.2 Bottom layer showing the vias which connect the ground plains both electrically and thermally. 
4.3 The actual TDM audio stream from the CS5368 chip, showing 8 channels. The yellow signal is SCLK, blue FS and green TDM serial stream. 
4.4 The actual TDM audio stream from the CS5368 chip, showing 1 channel. The yellow signal is SCLK, blue FS and green TDM serial stream. 
4.5 Noise on the 5V analog power rail. 
4.6 Noise on the 3.3V digital power rail. 
4.7 Noise on the 5V analog power rail when digital ground is used. 
4.8 Noise at the output of one ADC driver. The green and yellow are differential outputs and the purple one is the difference between yellow and green. 
4.9 The ADC module connected to the FPGA. The FPGA is programmed with the test code in appendix B.1. 
4.10 The TDM data stream is clocked into the FPGA and data ready (dr) signal goes high on the rising edge of each frame sync pulse (FS). 
4.11 The dr signal is 16 PCLK clock cycles long. 

A.1 Schematic diagram of the CS5368 chip and how it is connected. 
A.2 Schematic diagram of the ADC input driver. 
A.3 Schematic diagram of all ADC input drivers. 
A.4 Schematic diagram of $V_Q$ buffer and bypassing capacitor network. 
A.5 Schematic diagram of ATtiny167 MCU and header pins connections. 
A.6 Schematic diagram of the voltage regulators for analog and digital part. 
A.7 The PCB layout showing all layers. 
A.8 The PCB layout showing only top layer. 
A.9 The PCB layout showing only layer 2. 
A.10 The PCB layout showing only layer 3. 
A.11 The PCB layout showing only bottom layer.
List of Tables

2.1 Cirrus Logic CS5368 specification from datasheet. [8] ............................ 12
2.2 Requirements for ADC driver operational amplifier. ............................ 20
2.3 OPA1632 specification. [11] ................................................................. 20
2.4 LTC6362 specification. [12] ................................................................. 21
2.5 THS4521 specification. [10] ................................................................. 21
2.6 LP38693 specification. [14] ................................................................. 25

4.1 Power supply noise measurements. ..................................................... 47
4.2 ADC driver output noise measurements. .......................................... 49
4.3 CS5368 heat dissipation. ................................................................. 50
Introduction

Sound localization and beamforming is an interesting topic both regarding the mathematical theory and hardware design. Being able to focus a sound beam in certain direction, isolating a sound source as much as possible from other sound sources, is an ongoing research area because of its many applications. One important application is noise reduction. Sound localization involves finding the direction to a sound source with the highest amplitude or certain characteristics and then focusing a sound beam to that direction. This is for example used in conference systems when the direction of a speaker is found and a camera is focused to that direction. [1] To make this possible multiple microphones are needed to form a microphone array. The sound is sampled simultaneously from these microphones and signal processing performed with a suitable algorithm to form a sound beam or find the direction of a sound source. The number of microphones used and the type of microphone arrays depends on the application and the algorithms used. An example of a microphone array is the Eigenmike from mh acoustics, consisting of 32 microphones arranged in a sphere. [2] In figure 1 these microphones can be seen. A project in MIT included a microphone array consisting of 1020 microphones, possibly the largest ever built, to capture sound in extremely difficult environment where traditional methods are insufficient. [3]

Figure 1: The Eigenmike microphone from mh acoustics consists of a total of 32 microphones arranged in a sphere. [2]
Motivation

In all research and development it is vital to have some platform for testing and measurements. The aim of this project is to create a platform that is possible to use in development of algorithms and hardware for sound localization and beamforming systems. As mentioned at the outset microphone arrays are in many sizes and shapes depending on the application. It is not the goal to focus on any particular application in this project but to design a system that can easily be adapted to any research in this field. The initial motivation for this project was an idea from Baldur Þorgilson to create an external hearing aid with improved sound quality in noisy environments by using microphone arrays. There have been extensive research in using microphone arrays in this way and many ideas have been presented. For example one idea is to have microphone array embedded in eyeglasses. [4] A patent for an external microphone array connected to a hearing aid was recently filed. [5] And many other ideas would be worth mentioning. Using microphone array for this or other purposes is an interesting topic. The system that will be presented in this paper is intended to facilitate the development of such a system.

The research in sound localization and beamforming depends on the ability to sample audio from multiple microphones. It is of great benefit to have access to a system that can be easily extended as needed. For example when an algorithm for beamforming is being tested it facilitates progress if the size of the microphone array can be increased or decreased as needed and the data is accessible without trouble. Than the focus can be on the actual research, which is the algorithm, and not on solving hardware problems.
Organization

This project is divided into 5 chapters.

- Chapter 1 introduces the multi-channel microphone system. Three design topologies are shown and a topology is chosen for the system.

- Chapter 2 is detailed description of the design of the ADC module which is the microphone input module. The main components are chosen and reasons for those decisions are presented.

- Chapter 3 shows how the microphone data is to be streamed to a computer through USB 3.0 cable. The multiplexing method is described and a USB 3.0 module is chosen and reasons for that choice.

- Chapter 4 shows the results of the project. The ADC module will be analysed and tested and the FPGA programming results discussed.

- Chapter 5 concludes the project and discusses possible future work.
### Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Analog to Digital Converter</td>
</tr>
<tr>
<td>CC</td>
<td>Cross Correlation</td>
</tr>
<tr>
<td>DOA</td>
<td>Direction Of Arrival</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field-Programmable Gate Array</td>
</tr>
<tr>
<td>GCC</td>
<td>Generalized Cross Correlation</td>
</tr>
<tr>
<td>LRCK</td>
<td>Left/Right Clock</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>PHAT</td>
<td>Phase Transform</td>
</tr>
<tr>
<td>SCLK</td>
<td>Serial Clock</td>
</tr>
<tr>
<td>SRP</td>
<td>Steered Response Power</td>
</tr>
<tr>
<td>TDM</td>
<td>Time Division Multiplexing</td>
</tr>
<tr>
<td>TDOA</td>
<td>Time Difference Of Arrival</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
</tbody>
</table>
Chapter 1

Multi-channel microphone system

The goal of this project is to design a multi-channel microphone system that is flexible and easy to use. There are products on the market that could be used for this purposes. An example of that is product from Concurrent, CP-AD3224-DS, which is a 32 channel, 24 bit, Delta-sigma analog to digital conversion card for a desktop computer. [6] This card allows for easy sampling of multiple audio channels and the possibility of synchronizing multiple cards for higher channel count. A system like this is expensive and bulky. A desktop computer is needed and that could be disadvantage for example if the system is to be used in many different locations.

There are many solutions possible to create a multi-channel microphone system. The solutions chosen here is to design the system from scratch. One of the biggest advantages is that is allows for more flexibility in the design. Another advantage is the educational value of going through the whole design process.

1.1 System requirements

The most important decisions that need to be made about the system are the following:

1. What is the sample rate and bit depth?
2. How many microphone channels should be available?
3. Is simultaneous sampling needed?
4. How is the data to be sent to a computer?

This system is intended to be for research and development so it is reasonable to require high resolution sampling. Concerning the number of microphones it is important to decide how flexible the system should be. If the whole system is built around a certain number of channels it could be difficult to expand the system later if required. A high number of channels must be a requirement in a system like this
because it is not specified for a certain application. If compared to the Eigenmike, mentioned earlier, which has 32 microphones, some multiple of that number would be a good start. Concerning the data connection it must be kept in mind that the system should be easy to use for the end user and also be able to send data from a high number of microphones in real time. This would be best solved with an USB 3.0 connection. Than the number of microphones would only be limited to the maximum data rate of USB 3.0. After all these consideration the following requirements are made for the initial system:

1. Sound sampled at 192 kHz sample rate and 24-bits.
2. High quality sampling with low distortion.
3. Possible to sample from up to 128 microphones simultaneously.
4. Have individual modules that will take care of certain number of channels.
5. All channels should be sampled synchronously.
6. It should be easy to add microphones to the system as needed.
7. Send digital sound data from all channels in real-time through one USB 3.0 cable to a computer.
8. Small in size and good for research and educational use.

These requirements will be used throughout the design of the system. As mentioned before the system is intended for research purposes and will be subject to ongoing development. These requirements are considered to be a reasonable starting point for the design of the system.

1.2 Goals for initial design

Designing a complex system like this is subject to many uncertainties and it is impossible to know exactly how much can be done during this project. There are many problems that can delay the design process. The following goals are set for the project:

1. Make decision on the fundamental design of the system.
2. Decide how the data is to be transferred through USB.
3. Design the ADC module.
4. Get at least one ADC module to work as expected.
5. Stream audio data through USB 3.0.
1.3 Design topologies

When designing a system that can meet the above specification it is important to begin by making well thought out decisions on the overall working of the system. Many questions have to be addressed before considering the details of the design. This will help prevent major design changes later in the process which can both be expensive and time consuming. Before any attempts were made to tackle any specific design problems a design topology was carefully chosen. In the end three main topologies where considered and one of them chosen as the final system. Some of the important questions that needed to be answered where the following:

1. What kind of analog to digital converter is to be used?
2. What digital audio data protocol is to be used?
3. How to multiplex the audio data from all microphones?
4. How to stream the multiplexed data through one USB 3.0 cable?

There is no one obvious design topology that is the perfect solution to these problems. An extensive research was needed to find components and modules that would fit the design. The focus was first on finding a good analog to digital converter (ADC). The ADC that was chosen had an 8 channel input and a TDM output which clocks out all the audio data on to one wire. This component will be discussed in detail in section 2.1 and the reasons for choosing the TDM data format. To be able to have up to 128 microphone channels 16 of these ADC modules are needed. Another important issue was the multiplexing of all the data from different ADC. It was decided to use a field programmable gate array (FPGA) to multiplex the data. In section ?? the reasons for this choice will be discussed. There are two choices considered for USB 3.0 connection. Either to use a USB 3.0 chip and go through all the design issues related to that or use an USB 3.0 module which hides all most of the USB specific hardware and firmware issues. In section 3.2 the choice of USB module will be discussed.

1.3.1 Design topology 1

One possibility was to equip each ADC board with an USB connection. This would require an USB chip on each board. A block diagram of the solution can be seen on figure 1.1. All the ADC boards would be connected together through an USB hub. Although the data rate on each board would be well within the capabilities of USB 2.0 the total bandwidth from all 16 boards would have to be around 800 Mbps as was stated above and USB 3.0 is therefore a minimum requirement.
Chapter 1. Multi-channel microphone system

1.3. Design topologies

Using this design topology requires the following:

- Knowledge of the USB 3.0 protocol.
- Routing a high speed signal on a PCB.
- Creating a USB 3.0 driver for the system.

Each of these problems are time consuming and difficult to solve without special knowledge of all aspect of USB 3.0. The advantages of using this topology is that only an USB 3.0 hub is required to connect all the boards together.

1.3.2 Design topology 2

The second design was based on using the separate USB 3.0 module EZ-USB FX3 from Cypress (see section 3.2). The interface to this module is an 8, 16 or 32 bit parallel bus. The ADC sends serial data to the FPGA which converts the data to parallel data and stores it in a FIFO buffer. When data is ready to be read from the FPGA on all ADC modules it is clocked out to the USB module in predefined order. The data bus formed by the FPGA are tri-state so while data is being clocked out of one module the other modules are in high impedance state.
Chapter 1. Multi-channel microphone system

1.3. Design topologies

The main advantages of this design is that it has a slower bus and most of the USB 3.0 specific problems are not present in the design. There are though many disadvantages. Each module needs a data bus, an address bus and a bus control lines. This requires a high pin count on each ADC board and a lot of wires connecting the boards to the bus. The parallel port needs to run at a minimum clock frequency of 49.152 MHz so the data can be streamed in real time. This clock frequency can be calculated using the following formula:

\[ \frac{N_c \cdot C_b}{B_w} \cdot f_s = \frac{128 \cdot 32}{16} \cdot 192kHz = 49.152 MHz \]  \hspace{1cm} (1.1)

Where \( N_c \) is the number of channels, \( C_b \) number of bits per channel, \( B_w \) the bus width and \( f_s \) the sample rate. Although this is not a very high clock frequency compared to today’s standards, this can introduce many problems in this design. The amount of wiring needed for the parallel bus can cause a crosstalk between wires, timing issues between individual bits and connection problems due to bad wires or cold soldering. This is considered to be the main disadvantage to this topology.

1.3.3 Design topology 3

The third design topology is in part based on the second topology but with some important changes. The EZ-USB FX3 USB 3.0 module is used and a 16 bit data bus. But instead of placing an FPGA on each ADC board, only one FPGA is used. The serial data from all ADC boards are all streamed into this one FPGA. There the data is parallelized and all data streams have their dedicated FIFO buffers. The audio data is
then clocked from the FPGA to a 16 bit bus and into EZ-USB FX3 module.

![Diagram of design topology 3](image.png)

Figure 1.3: Design topology 3.

There are many advantages of this topology. There is still a 16 bit wide parallel bus but this bus has much less wiring because only two modules are connected to it instead of seventeen modules, the sixteen ADC modules and the EZ-USB FX3 module. The parallel bus can have only short wires which further reduces the problems associated with parallel buses. Another advantage is that only one FPGA is needed instead of sixteen which reduces cost dramatically. The main downside of this method is that a more complicated FPGA programming is needed.

### 1.3.4 Choosing a design topology

After carefully considering the advantages and disadvantages of each design topology design topology 3 was chosen for this project. The main reasons for that choice can be summed up as follows:

- USB 3.0 specific problems almost entirely avoided.
- Only two modules connected to parallel bus.
- One FPGA instead of 16.
- All FIFO buffers and multiplexing of channels are taken care of in one FPGA.
- Audio data from all ADC modules is streamed simultaneously into one FPGA which makes FPGA programming easier.

In chapter 2 the design of the ADC module will be discussed. In chapter 3 the method on streaming the audio data to a PC will be shown.
Chapter 2

ADC module design

In this chapter the design process of the ADC module of the microphone system will be discussed. There are many factors that need to be considered when designing a complex system as this multi-channel microphone system. The design has to be done in a specific order to make sense. For example it is not possible to accurately calculate the power consumption of the system and find adequate regulators before decision have been made about other components. The design process was done in the following order:

1. Analog to digital converter (ADC).
2. Input drivers for ADC.
3. Voltage regulators.
4. USB 3.0 module chosen.
5. FPGA module chosen.

2.1 Analog to Digital Converter

The most important component of the design is the analog to digital converter. As stated in the beginning of this chapter a high quality sampling was required at 192 kHz and with 24 bit conversion. Due to the high number of channels it was considered best to have an ADC with high number of inputs. The reason for this is to limit the number of chips on each ADC module and also the number of modules needed to get the 128 microphone channels required. The requirements made for the ADC were the following:

1. 8 channels with differential inputs.
2. Minimum of 192 kHz sampling rate.
3. 24 bit conversion.

4. Simultaneous sampling.

5. Delta sigma converter.

After an extensive search it was obvious that audio ADC with these specifications are not very common. Only ADC’s for high end audio recording equipment meets those specification. One ADC found that was of great interest is the CS5368 from Cirrus Logic. This ADC meets all the above specification. Cirrus Logic is a premium supplier of audio IC’s. [7] Although other suppliers like Texas Instruments have many high quality ADC for audio equipment those ADC’s did not meet the requirements of channel count, simultaneous sampling or sampling rate. The CS5368 specification can be seen in table 2.1.

<table>
<thead>
<tr>
<th>CS5368 specification:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Advanced Multi-bit Delta-Sigma Architecture</td>
<td></td>
</tr>
<tr>
<td>24-Bit Conversion</td>
<td></td>
</tr>
<tr>
<td>114 dB Dynamic Range</td>
<td></td>
</tr>
<tr>
<td>-105 dB THD+N</td>
<td></td>
</tr>
<tr>
<td>Supports Audio Sample Rates up to 216 kHz</td>
<td></td>
</tr>
<tr>
<td>Selectable Audio Interface Formats</td>
<td></td>
</tr>
<tr>
<td>– Left-Justified, I2S, TDM</td>
<td></td>
</tr>
<tr>
<td>– 8-Channel TDM Interface Formats</td>
<td></td>
</tr>
<tr>
<td>Low Latency Digital Filter</td>
<td></td>
</tr>
<tr>
<td>Less than 680 mW Power Consumption</td>
<td></td>
</tr>
<tr>
<td>On-Chip Oscillator Driver</td>
<td></td>
</tr>
<tr>
<td>Operation as System Clock Master or Slave</td>
<td></td>
</tr>
<tr>
<td>Auto-Detect Speed in Slave Mode</td>
<td></td>
</tr>
<tr>
<td>Differential Analog Architecture</td>
<td></td>
</tr>
<tr>
<td>Separate 1.8 V to 5 V Logic Supplies for Control and Serial Ports</td>
<td></td>
</tr>
<tr>
<td>High-Pass Filter for DC Offset Calibration</td>
<td></td>
</tr>
<tr>
<td>Overflow Detection</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: Cirrus Logic CS5368 specification from datasheet. [8]

As can be seen from table 2.1 the CS5368 meets the above requirements and is considered to be an excellent choice for this project. In figure 2.1 a basic block diagram of CS5368 is shown.
Figure 2.1: Block diagram of the CS5368 ADC. [8]

The block diagram of the CS5368 chip shows clearly the simultaneous sampling capabilities whereas the 8 differential inputs all have their dedicated Delta-sigma ADC. Also included in the chip is decimation filter and a high pass filter. The decimation filter makes an external anti-aliasing filter unnecessary. The control of the CS5368 can either be performed by control pins or by changing register values through SPI or I²C serial interfaces. It was decided to use the SPI interface to control the CS5368. To do that an external microcontroller (MCU) is used. In section 2.3 the choice of MCU is discussed. There are several audio interfaces available on the CS5368. These are left-justified (LJ), I²S and TDM. These interfaces will be discussed in the next three sections and than one of these interfaces will be chosen.

2.1.1 Left-Justified serial audio format

The left-justified format can be seen on figure 2.2. This format is a serial audio stream of two audio channels. Four channels are needed for the 8 channels in CS5368. Three signals are needed but two are shared between LJ channels, the serial clock (SCLK) and Left/Right Clock (LRCK) signal which is used to synchronize the left/right channel in a two channel system. When LRCK is high odd channel is being clocked out and when low, even channel is being clocked out. The number of bits per channel is 32 if the sample size is 24 bits. The remaining 8 bits are set to 0. As the name suggests, left-justified format justifies the sample to the left of LRCK pulse so the extra 8 zero bits are to the right. The MSB bit is clocked out first and is coincident with the leading transitions in the LRCK signal. [9]
2.1.2 I²S serial audio format

The I²S has many things in common with the LJ audio format as can be seen by comparing figures 2.2 and 2.3. Both formats are for two audio channels and the CS5368 has four I²S outputs to clock out all 8 channels. The main difference between I²S and LJ format is that in I²S the odd numbered channels are clocked out when LRCK is low and even channels when LRCK is high. Another difference is that the MSB is one SCLK period after a transition of the LRCK signal.

2.1.3 TDM serial audio format

The TDM (Time Division Multiplexing) audio format can be viewed as an extension of the I²S format. Instead of only clocking out two channels during one LRCK cycle an N number of channels can be clocked out during a signal called frame sync (FS). All 8 audio channels of the CS5368 can thus be contained in one serial data stream. This can be seen on figure 2.4.
2.1.4 Choosing the serial audio format

The serial audio format chosen for the multi-channel microphone system was the TDM format. The main reason for that choice was to make the overall system complexity lower due to lower signal count. Instead of having four serial audio streams along with the SCLK and LRCK signals in I²S and LJ formats there is only one serial stream along with the SCLK and FS signals in TDM format. In section 1.3 a design topology was chosen which includes using an FPGA. If the I²S format would be used the input pins needed for all the serial audio data of the 16 ADC modules are 66 inputs. The TDM format only requires 18 input pins for the serial audio data. It is obvious that system complexity and cost are reduced by using the TDM format.

2.1.5 CS5368 pin and connection diagram

Figure 2.5 shows the pins of the CS5368 and how it is connected according to the datasheet for the chip. This diagram will be used as a reference later in the design process.
2.1.6 CS5368 clock sources and sampling synchronization

The CS5368 needs an external clock source to do the analog to digital conversion and operate the internal digital circuitry. There are three clock options for the CS5368:

1. Master mode: Oscillator tank circuit connected to on-board driver through XTI and XTO pins.

2. Master mode: External clock connected to MCLK pin with XTI and XTO pins grounded.

3. Slave mode: External clock connected to SCLK and LRCK/FS pins.

In master mode the SCLK and LRCK/FS pins are outputs for the serial clock and sync clock respectively. An high precision and stable crystal is required to get
Chapter 2. ADC module design 2.1. Analog to Digital Converter

an accurate sampling frequency or some other clock source connected to MCLK pin meeting those requirement.

When the intention is to use more than one CS5368 to sample synchronously it is important to synchronize the clock of all CS5368. In this multi-channel microphone system 16 CS5368 are needed and all chips must sample the 128 channels synchronously. This is made possible by configuring one CS5368 in master mode and all other in slave mode and connect SCLK and LRCK/FS pins together. This can be seen on figure 2.6.

![Figure 2.6: Synchronizing the sampling of many CS5368.](image)

2.1.7 CS5368 clock frequency

To sample the standard sampling rate of 192 kHz a master clock of 49.152 MHz is needed. There are internal clock dividers in the CS5368 to get sample rates of 96 kHz and 48 kHz. This is done by changing the MDIV bits in GCTL register through SPI. Therefore with a master clock frequency of 49.152 MHz three standard sampling frequencies can be selected. The CS5368 datasheet [8] is somewhat misleading about the master clock frequency. The datasheet might be understood that a crystal of 12.288 MHz would give a master clock frequency of 49.152 MHz with PLL. This was understood to be the case in the design process. To get a sample frequency of 192 kHz a 49.152 MHz is needed or an external clock signal with the same frequency on MCLK pin with XTI and XTO pins grounded.

2.1.8 CS5368 reference voltage

The reference voltage for the internal sampling circuit is the analog voltage on pin VA. The CS5368 needs a 5V analog voltage. FILT+ pin outputs the reference voltage and
should be filtered with a capacitor as stated in the CS5368 datasheet. The VQ output pin is internal quiescent reference voltage. The output voltage of this pin is $V_A/2\,\text{V}$ where $V_A$ is the analog operating voltage. The output from this pin can be used as the offset voltage for the input drivers, discussed in section 2.2. Generally this reference voltage has the label $V_{\text{OCM}}$.

### 2.1.9 CS5368 DC power consumption

The power consumption for the CS5368 is shown in figure 2.7 which is a table taken from CS5368 datasheet. [8] The maximum power consumption is 792 mW when a 3.3 V is used for $V_D$, $V_{LS}$ and $V_{LC}$. This power consumption is when the master clock is at 12.288 MHz. As was stated in section 2.1.7 the datasheet for CS5368 was somewhat misleading regarding the clock frequency. A master clock frequency of 49.152 MHz is needed for a sampling rate of 192 kHz. When calculating the power supply current the numbers in the table in figure 2.7 where used which is in fact much less current than is needed at a sampling rate of 192 kHz.

![Figure 2.7: Table from CS5368 datasheet showing DC power consumption.](image)

From the above table from CS5368 datasheet it is not clear how much current the module will consume at a sample rate of 192 kHz. There are no information about how the current changes proportionally to the master clock frequency.

### 2.2 ADC input drivers

An important factor in the design are input drivers for the CS5368 ADC. These drivers isolate the driver Cirrus Logic recommends for the CS5368 is shown on figure 2.8. This driver has two operational amplifiers with single ended outputs. By combining these drivers as shown a differential output is formed. It has become increasingly common in later years to use operational amplifiers with differential outputs for applications...
like driving ADCs with differential inputs. It was decided to use operational amplifier with differential outputs to drive the inputs of CS5368. In figure 2.9 an example of ADC input driver with differential outputs can be seen.

![Diagram of ADC input driver with differential outputs](image)

Figure 2.8: Input driver for the CS5368 ADC as suggested by Cirrus Logic in datasheet. [8]

![Diagram of another ADC input driver](image)

Figure 2.9: An example of ADC driver with differential outputs using THS4521 operational amplifier form Texas Instrument. [10]

When choosing an operational amplifier for the input drivers the requirements in
Chapter 2. ADC module design

2.2. ADC input drivers

table 2.2 had to be met. Some of these requirements are of general nature.

<table>
<thead>
<tr>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low distortion.</td>
</tr>
<tr>
<td>Low noise.</td>
</tr>
<tr>
<td>Fully differential input and output.</td>
</tr>
<tr>
<td>Rail-to-rail inputs and outputs.</td>
</tr>
<tr>
<td>Fully Differential Input and Output</td>
</tr>
<tr>
<td>Single supply.</td>
</tr>
<tr>
<td>High gain bandwidth product.</td>
</tr>
</tbody>
</table>

Table 2.2: Requirements for ADC driver operational amplifier.

Three choices of operational amplifiers will be presented. The first amplifier considered was the Texas Instruments OPA1632. As has been stated the aim was to build a multi-channel microphone system of high quality. The OPA1632 amplifier is of exceptional quality for audio applications. The OPA1632 specification are in table 2.3. This amplifier was initially chosen for this design because of its exceptional quality. However, when it was discovered that it did not feature a rail-to-rail inputs and outputs it was decided not to use the OPA1632. Supplying it from the 5 V analog supply voltage would have limited the dynamic input range dramatically and having a positive and negative supplies was not considered a good choice due to increased system complexity and cost. Another downside is the high price tag.

<table>
<thead>
<tr>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distortion: 0.000022%</td>
</tr>
<tr>
<td>Noise: 1.3 nV/$\sqrt{Hz}$</td>
</tr>
<tr>
<td>Slew Rate: 50 V/$\mu$s</td>
</tr>
<tr>
<td>Gain Bandwidth: 180 MHz</td>
</tr>
<tr>
<td>Fully Differential Input and Output</td>
</tr>
<tr>
<td>Supply Range: ±2.5 V to ±16 V</td>
</tr>
<tr>
<td>Rail-to-rail Inputs and Outputs: No</td>
</tr>
</tbody>
</table>

Table 2.3: OPA1632 specification. [11]

Another operational amplifier considered was the LTC6362 from Linear Technology. Some of the important specifications are listed in table 2.4. It is not straightforward to compare the distortion values of the LTC6362 and OPA1632 because of different approaches used but in short the OPA1632 has a better distortion values. Although the LTC6362 operational amplifier is not of the same quality as the OPA1632 it meets all the requirements listed in table 2.2 and was thus chosen for the ADC module. This part
was available at the time it was chosen but was out of stock when the order was being made. Therefore it was not possible to use LTC6362 and another part was chosen, the THS4521 from Texas Instruments.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distortion: –116dBc at 1kHz, 8V_{P-P}</td>
<td></td>
</tr>
<tr>
<td>Noise:</td>
<td>3.9 nV/√Hz</td>
</tr>
<tr>
<td>Slew Rate:</td>
<td>45 V/µs</td>
</tr>
<tr>
<td>Gain Bandwidth:</td>
<td>180 MHz</td>
</tr>
<tr>
<td>Fully Differential Input and Output</td>
<td></td>
</tr>
<tr>
<td>Supply Range:</td>
<td>2.8 V to 5.25 V</td>
</tr>
<tr>
<td>Rail-to-Rail Inputs and Outputs:</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 2.4: LTC6362 specification. [12]

Some of the specification of the THS4521 operational amplifier are in table 2.5. The specification are similar to that of the LTC6362. The THS4521 has a higher noise level and more distortion. This is of course a downside but this is still an operational amplifier of good quality and well suited for the ADC module.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distortion: –112dBc (0.00025%) at 1kHz 5V_{P-P}</td>
<td></td>
</tr>
<tr>
<td>Noise:</td>
<td>4.6 nV/√Hz</td>
</tr>
<tr>
<td>Slew Rate:</td>
<td>490 V/µs</td>
</tr>
<tr>
<td>Gain Bandwidth:</td>
<td>180 MHz</td>
</tr>
<tr>
<td>Fully Differential Input and Output</td>
<td></td>
</tr>
<tr>
<td>Supply Range:</td>
<td>±2.5 V to ±16 V</td>
</tr>
<tr>
<td>Rail-to-rail Inputs and Outputs:</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 2.5: THS4521 specification. [10]

The input driver on figure 2.8 as shown in Cirrus Logic datasheet for the CS5368 ADC [8] was used as a reference design for the input driver for the ADC module. The schematic can be seen on figure 2.10. The same resistor and capacitor values are used. The only change is that instead of using two operational amplifiers with single ended output, one amplifier is used with differential output. The input driver as shown in the datasheet is designed with the master clock frequency of 12.288 MHz in mind. As was stated in section 2.1.7 a master clock frequency of 49.152 MHz is needed to sample at 192 kHz. This amplifier is designed to attenuate noise at 6.144 MHz, half the master clock frequency of 12.288 MHz. The reason for this filter is that the digital filter in the CS5368 does not reject input signals of $N \times 6.144$ which is the digital passband.
frequency. [8] Even if the master clock frequency would be 49.152 MHz and the digital passband frequency 24.576 MHZ the input amplifier will filter out that frequency. In the reference design there are 10\(\mu\)F input capacitors. These capacitors form a high pass filter in conjunction with the 10 k\(\Omega\) resistors with the 3 dB corner frequency at approximately 1.59 Hz as described in Cirrus Logic application notes AN241. [13] The 10 k\(\Omega\) resistors are not present in the final design and were incorrectly included without proper calculations. This matter will need to be clarified in future work on the ADC module.

![Figure 2.10: The input driver for the ADC module using the THS4521 operational amplifier.](image)

### 2.3 MCU for ADC module

The MCU for the ADC module is intended to configure the CS5368 through SPI and open the possibility of accessing all modules through one serial link. This is made possible by connecting all ADC modules to a I\(^2\)C bus and have one master module with UART connection. In view of this the MCU has to have the following specification:

1. Internal oscillator.
2. SPI, UART, I\(^2\)C serial communication.
3. Can operate at 3.3 V.
4. At least 16 I/O pins.

5. Small package size.

It was decided to use MCU from Atmel with ARM architecture. That would open the possibility of using the Arduino programming environment. That was considered a good choice in this project because only a simple program is needed for the MCU and it is easily implemented in the Arduino environment. The MCU chosen was the ATtiny167. This MCU meets all the above requirements. It is available in QFN32 package which suits well for the ADC module due to its small size. The initial code for the ATtiny167 can be seen on figure 2.11. The only thing being the MCU does is to write configuration bits to register GCTL at address 0x01.
Chapter 2. ADC module design 2.4. Voltage regulators

It was decided to use two voltage regulators for the ADC module, one 5 V regulator for the analog part and 3.3 V regulator for the digital part. Running the analog and digital part on separate voltage regulators is encouraged in the CS5368 datasheet due to noise consideration. [8] As can be seen on figure 2.7 the power consumption is less in the digital part of the circuit when 3.3 V are used. For the analog part the only

```c
#include <SPI.h>
//#include <tinySPI.h>
void sleepNow();
bool start;
void setup() {
  pinMode(PB6, OUTPUT);
  pinMode(PB7, OUTPUT);
  digitalWrite(PB6, HIGH);
  digitalWrite(PB7, LOW);
  SPI.begin();

  start = true;
}

void loop() {
  delay(100);
  digitalWrite(PB7, HIGH);
  delay(400);
  if (start) {
    //pinMode(SCK, OUTPUT);
    //digitalWrite(SCK, HIGH);
    digitalWrite(PB6, LOW); //Chip select low to start SPI transfer
    //pinMode(SCK, OUTPUT);
    //digitalWrite(SCK, HIGH);
    SPI.begin();
    SPI.transfer(0b10011110); //CS5368 address with write enabled.
    SPI.transfer(0x01); //Register address.
    SPI.transfer(0b10001010); //Value to write to register

    SPI.end();
    digitalWrite(PB6, HIGH); //Chip select high to stop transfer.
    pinMode(SCK, OUTPUT);
    digitalWrite(SCK, LOW);

    digitalWrite(PB7, LOW);
    delay(100);
    digitalWrite(PB7, HIGH);

  }
}
```

Figure 2.11: The program for ATtiny167 MCU on ADC module

2.4 Voltage regulators
supported voltage is 5 V. Only linear regulators are considered for this design. The reasons for that should be quite obvious. Switching regulators, although extremely power efficient, naturally produce a lot of noise into the circuit which can be hard to filter out. The digital part of the circuit is immune to this noise but the analog part can suffer dramatically and that will degrades the quality of the whole system.

Power efficiency was not the main focus when choosing regulators because the system is not intended to be driven by batteries. Heat dissipation is of concern but that can be regulated to some extent by minimizing the maximum allowed input voltage. The requirements made for the voltage regulators are shown in table

1. Available in SMD package with thermal pad.
2. Supply currents of up to 400 mA for both 5V and 3.3V power rails.
3. Dissipate heat to the ground plane of the PCB.
4. Low junction-to-board thermal resistance.
5. High output accuracy.
6. Good ripple rejection characteristics.

There are many linear voltage regulator on the market that meet these requirements. After a search on Digi-Key website (digikey.com) a voltage regulators from Texas Instruments was chosen with the part number LP38693. The most relevant specification are shown in table 2.6 but more details are in the part datasheet. [14]

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum input voltage</td>
<td>10 V</td>
</tr>
<tr>
<td>Accuracy</td>
<td>2%</td>
</tr>
<tr>
<td>Dropout voltage</td>
<td>250 mV at 500 mA</td>
</tr>
<tr>
<td>Output current</td>
<td>500 mA (Internally limited)</td>
</tr>
<tr>
<td>Ripple rejection</td>
<td>55 dB</td>
</tr>
<tr>
<td>Junction-to-board thermal resistance</td>
<td>13.0 °C/W</td>
</tr>
<tr>
<td>Thermal overload protection</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.6: LP38693 specification. [14]

The LP38693 voltage regulator comes in three packages, 3-pin TO-252, 5-pin SOT-223 and 6-pin WSON with exposed thermal pad. In the datasheed for the LP38693 it can be seen that the SOT-223 package has the lowest junction-to-board thermal resistance of 13.0 °C/W. The SOT-223 package can be seen on figure 2.12. This package was chosen due to its low junction-to-board thermal resistance and small size.
Chapter 2. ADC module design

2.4. Voltage regulators

Figure 2.12: SOT223 package used for the LP38693 voltage regulator. [14]

2.4.1 Thermal resistance and input voltage

It is important to find the maximum allowed input voltage for the ADC module to regulate the heat dissipation of the voltage regulators. The LP38693 have a thermal overload protection which means that they shut down when certain temperature is reached. According to the datasheet this threshold is set at 160 °C. It was decided to allow the regulators to reach 90 °C which is well below the maximum temperature of 150 °C according to the datasheet. The power dissipated in the regulators are found by the equation:

\[ P_{\text{Reg}} = V_{\text{Reg}} \cdot I_{\text{Total}} \] (2.1)

Where \( P_{\text{Reg}} \) is the power dissipated in the regulator and \( V_{\text{Reg}} \) the voltage drop over the regulator and \( I_{\text{Total}} \) the total current. The power dissipated in the regulator is inevitably transferred into heat. The initial approach to calculate the maximum allowed voltage drop was using the following equation which was later found to be incorrect as explained later:

\[ P_{\text{Reg}} = \frac{T_J - T_A}{R_{\Theta JB}} \] (2.2)

Where \( T_J \) is the junction temperature, \( T_A \) the ambient temperature and \( R_{\Theta JB} \) the junction-to-board thermal resistance. By combining equations 2.1 and 2.2 we get:

\[ V_{\text{Reg}} = \frac{T_J - T_A}{R_{\Theta JB} \cdot I_{\text{Total}}} \] (2.3)

By using this equation it is possible to calculate the maximum allowed voltage drop given the total current, ambient temperature which was chosen to be 25 °C, allowed junction temperature and thermal resistance.

\[ V_{\text{Reg}} = \frac{T_J - T_A}{R_{\Theta JB} \cdot I_{\text{Total}}} = \frac{90 - 25}{13.0 \cdot 0.400} = 12.5V \] (2.4)
According to these calculations the maximum allowed input voltage is found by using the 3.3 V regulator and therefore it is:

\[ V_{\text{In, max}} = 3.3 + 12.5 = 15.8 \text{V} \]  \hspace{1cm} (2.5)

This means that a total of

\[ P_{\text{Reg, 3V3}} = V_{\text{Reg, 3V3}} \cdot I_{\text{Total}} = 12.5 \cdot 0.400 = 5 \text{W} \]  \hspace{1cm} (2.6)

are dissipated in the 3.3 V regulator. A maximum input voltage of 12 V was decided to make some headroom for error. But as stated earlier these calculations are not correct. First it was overlooked in the design that the maximum allowed input voltage for LP38693 regulator is 10 V. Another problem is that the junction-to-board thermal resistance does not account for the board-to-ambient thermal resistance. Other components on the PCB also dissipate heat and would need to be accounted for in the calculations. An ambient temperature of 25 °C is also low if the ADC module is not contained in a ventilated area. An ambient temperature of 40 °C is thus considered to be a safer approach. Equation 2.2 can be extended to include the board-to-ambient thermal resistance:

\[ P_{\text{Reg}} = \frac{T_J - T_A}{R_{\Theta JB} + R_{\Theta BA}} = \frac{T_J - T_A}{R_{\Theta JA}} \]  \hspace{1cm} (2.7)

By using equation 2.7 and solving it for \( R_{\Theta JA} \) to get the total thermal resistant needed for the 3.3 V regulator for a 12 V input voltage we get:

\[ R_{\Theta JA} = \frac{T_J - T_A}{P_{\text{Reg}}} = \frac{T_J - T_A}{(V_{\text{In, max}} - V_{\text{Reg, 3V3}}) \cdot I_{\text{Total}}} = \frac{90 - 90}{(12 - 3.3) \cdot 0.4} = 14.4 \text{°C/W} \]  \hspace{1cm} (2.8)

These calculations show that the board-to-ambient thermal resistance needs to be \( R_{\Theta JA} - R_{\Theta JB} = 14.4 - 13.0 = 1.4 \text{°C/W} \). By using guidelines from Texas Instruments [15] it can be shown that such a low board-to-ambient thermal resistance is not possible in this design. In these guidelines the following formula is given to get the minimum board size for a given \( R_{\Theta JA} \) and \( R_{\Theta JB} \):

\[ \text{Board area (cm}^2) \geq \frac{500 \text{°C} \cdot \text{cm}^2}{R_{\Theta JA} - R_{\Theta JB}} = \frac{500 \text{°C} \cdot \text{cm}^2}{R_{\Theta BA}} = \frac{500 \text{°C} \cdot \text{cm}^2}{14.4 - 13.0} = 357 \text{cm}^2 \]  \hspace{1cm} (2.9)

The total board size of the final design is 61.56 cm\(^2\) with 4 layers of copper as will be discussed in section 2.5. If top and bottom layers could be fully utilized the area is around 120 cm\(^2\) which is of course not the case. A total area of 15 cm\(^2\) will be used for the calculations for each layer. Four vias where added close to the regulators as can be
seen on figures 2.13 and 2.14 and they add thermal resistance that needs to be added correctly to the junction-to-ambient thermal resistance. The thermal resistance of the copper layers can be found by using equation 2.9 with respect to $R_{\Theta_{BA}}$ and using an area of 15 cm$^2$ for each layer:

$$R_{\Theta_{BA}} = \frac{500^\circ C \times \text{cm}^2}{W} = \frac{500^\circ C \times \text{cm}^2}{15} = 33.3^\circ C/W$$ \hspace{1cm} (2.10)

Using the guidelines from Texas Instrument the thermal resistance for the four vias can be found with the equation [15]:

$$R_{\Theta_{VIA}} = \frac{261^\circ C \times \text{cm}}{W} = \frac{261^\circ C \times \text{cm}}{4} = 65.3^\circ C/W$$ \hspace{1cm} (2.11)

The vias thermally connect the top and bottom layer so the thermal resistance of the vias and bottom layer are added together.
\[ R_{\Theta V IAS, bottom} = R_{\Theta V IAS} + R_{\Theta BA, bottom} = 65.3 + 33.3 = 98.6^\circ C/W \]

The top layer thermal resistance and \( R_{\Theta V IAS, bottom} \) are in parallel and the total board-to-ambient thermal resistance can be calculated by [16]

\[
R_{\Theta BA, total} = \frac{1}{\frac{1}{R_{\Theta BA, top}} + \frac{1}{R_{\Theta V IAS, bottom}}} = \frac{1}{\frac{1}{33.3} + \frac{1}{98.6}} = 24.9^\circ C/W \quad (2.12)
\]

The total thermal resistance given the above surface area is therefore

\[ R_{\Theta JA} = R_{\Theta JB} + R_{\Theta BA, total} = 13.0 + 24.9 = 37.9^\circ C/W \]

The maximum voltage drop over the LP38693 3.3 V regulator can thus be calculated as

\[
V_{Reg3V3} = \frac{T_J - T_A}{R_{\Theta JA} \cdot I_{Total}} = \frac{90 - 40}{37.9 \cdot 0.4} = 3.3V \quad (2.13)
\]

And thus a maximum input voltage of

\[ V_{In, Max} = V_{Reg3V3} + 3.3 = 3.3 + 3.3 = 6.6V \]

is allowed for the 3.3 V regulator. The 5 V regulator would therefore dissipate a total of

\[ P_{D5V} = (V_{In, Max} - V_{Reg5V}) \cdot I_{Total} = (6.6 - 5.0) \cdot 0.4 = 1.6 \cdot 0.4 = 0.64W \]

The same calculations apply to the thermal resistance for the 5 V regulator and therefore this is well below the maximum allowed power dissipation. The maximum allowed input voltage was decided to be 6 V instead of 6.6 V. This further lowers the heat dissipation of the regulators which is always a better option.

### 2.5 PCB layout and noise filtering

The design of the ADC module can be found in the appendices, the schematic in appendix A.1 and PCB layout in appendix A.2. The software used in the design was EAGLE from CadSoft. Although the well known Altium would have been preferred, Eagle is lighter and has previously been used with good results by the author of this paper. It was decided to use a 4 layer PCB which allows for more compact design and easier signal routing. The PCB was manufactured in China by a company found on eBay.com. The EAGLE files where converted to a gerber format and sent to the
manufacturer. The manufactured PCB can be seen on figures 2.15 and 2.16. These PCB are of good quality and worked as expected.

Figure 2.15: Top layer of the PCB for ADC module.

Figure 2.16: Bottom layer of the PCB for ADC module.

2.5.1 Power rail noise filtering

Good filtering of the power rails is an important way of reducing the noise in the circuit. To filter high frequency noise ceramic bypass capacitors are used as shown on figure 2.17. This filter was used in many parts of the PCB. In figure 4.2 these filters can be seen as pairs of SMD ceramic capacitors. Theoretically the 1 µF capacitor should be able to bypass all the frequencies the 0.01 µF bypasses. In practice all capacitors also
have a parasitic inductance and resistance and are in fact a RLC circuit. As a result the measured frequency response of a capacitor is different from that of theoretical frequency response. Ceramic capacitors have a V shaped response curve. [17]

![Figure 2.17: Filter for the 5 V analog power rail.](image)

Figures 2.18 and 2.19 shows the frequency response of the TDK 0.01\(\mu\)F and 1\(\mu\)F capacitors used in the design. As can be seen the response is not as of an ideal capacitor but as a bandpass filter. At lower frequencies the capacitor behaves similar to an ideal one but at higher frequencies the inductor starts to have more impact on the response. The lowest impedance is at the resonance frequency of the RLC circuit which is naturally not the same for each capacitor because of different RLC values. By having pairs of ceramic capacitors with different resonance frequency it is possible to have a better bypassing over wider frequency band. This was used in the design of the ADC module and therefore a pair of 0.01\(\mu\)F and 1\(\mu\)F capacitors are used to filter the power rail.

![Figure 2.18: Frequency response of the TDK 0.01\(\mu\)F capacitor.](image)
2.5.2 PCB layout and noise

The layout and routing of a PCB has a great impact on the quality of the overall system. The ADC module is a mixed-signal circuit. Analog signals are very sensitive to noise because the noise will become part of the information included in the signal and lowers the signal to noise ratio (SNR). The information content of a digital signal only changes due to noise when the noise level is so high that individual bits are difficult to distinguish and is therefore more tolerant to noise. The digital part of a circuit produces high frequency noise which can contaminate the analog signals. It is thus important to design the PCB in such a way to minimize the noise from the digital part. This is done partly with a good filtering of the power rails with bypass capacitors as described in section 2.5.1. These capacitors can be seen in pairs on figure 4.2. They are placed under every input driver operational amplifier to filter the supply voltage as close to the power pins as possible. Bypassing capacitors are though not the only design methods used to minimize noise in the analog part. Placement of individual components, ground plane layout and routing of signals are also important to consider.

In a mixed signal PCB the ground plane layout is especially important. The return currents of all signals on a PCB takes the route of minimum impedance through the ground connection, preferably under the signal trace. This can be seen on figure 2.20. A ground plane is a low-impedance return path for high-frequency currents and it is important to be aware of the actual return current path of individual signal, not only the signal path. In a mixed signal PCB design the return currents of a high-frequency signal can otherwise interfere with other signals. The ground plane in a mixed signal design should therefore be partitioned and connected at a common point. All signal traces passing between the analog and digital part should be routed through that point, as can be seen on figure 2.21. [22]
When designing the PCB for the ADC module the analog and digital ground planes were connected at a common point in accordance with the above design guidelines. This can be seen on figure 2.22 which shows where the ground planes are separated and the common connection point. All return currents from the digital components are thus kept away from analog signal traces and interference from high-frequency signals should be minimized. One of the advantage of using an ADC converter with differential inputs is that the signal does not need the ground for its return current. The return current flows through the other trace. [22] It is important to route differential signals close together so that noise adds to the differential signal pair equally. At the receiving end common noise will cancel out to a large extent. As can be seen on figures A.8 to A.11 the differential signal pairs were kept close together for these reasons.
Figure 2.22: The PCB for the ADC module has separate ground for the digital part and analog part, only connecting at a common point (red circle). No return currents from components producing high-frequency signals should have to pass through the analog part.
Chapter 3

Streaming audio data to PC

In section 1.3 a design topology was chosen for the multi-channel microphone system. There a FPGA and USB modules where introduced which had not yet been described in detail and the reasons for these choices. From the beginning it was a requirement that all the audio data would be streamed through one USB 3.0 cable. It was considered to be the most convenient way of streaming the data. USB 3.0 is easy to use for the end user and has the required data rate for this application as will be shown in section 3.1. The audio serial data from the 128 microphone channels are performed in 16 different ADC modules. This data needs to be multiplexed to create one serial audio data stream for the USB cable. There are two methods considered:

1. FPGA for multiplexing and a separate USB 3.0 module.

2. Processor to handle both the multiplexing and USB 3.0 connection.

The second option is not considered a viable option. It would of course be convenient to encapsulate the whole process in one signal processor with USB 3.0 interface. The problem with that though is that 16 TDM serial inputs are needed which all have to work in real time. The multiplexing would have to be done through software in the processor by implementing FIFO buffers and probably by using DMA where data would be streamed directly from memory locations to the USB cable. Another problem is that audio TDM stream has not been standardize and there can be variations between formats. [23] The I²S format is on the other hand standardized and many processors include I²S port. But to stream 128 microphone channels 64 I²S ports are needed because it is a 2 channel protocol. With this in mind it was not considered a viable option to use a processor to multiplex the microphone data. An approach using an FPGA and a separate USB 3.0 module was chosen instead. As will be explained in section 3.3 FPGAs are well suited for multiplexing such a high number of serial data streams in real time. Choosing a suitable USB 3.0 module will be discussed in section 3.2.
Chapter 3. Streaming audio data to PC

3.1 System data rate

As has been stated the required sample rate and bit depth of each microphone channel is 192 kHz and 24 bits, respectively. In the TDM data stream one sample is 32 bits with the 8 extra bits set to zero. The raw data bit rate of the system is therefore

\[ R_{b,TDM} = N_{ch} \cdot N_{b,ch} \cdot F_s = 128 \cdot 32 \cdot 192000 = 786,432,000 \text{ bit/s} \]

Where \( N_{ch} \) is the number of channels in the system, \( N_{b,ch} \) the number of bits per channel and \( F_s \) the sample rate. For the system to work in real time without any loss of data the minimum system throughput of sampled data needs to be 786,432,000 bit/s.

3.2 USB 3.0 module

The data that needs to be transferred is 786.432 Mbps but USB 2.0 has a bandwidth of only 480 Mbps. [24] To transfer all these data using one cable USB 3.0 is chosen which as a maximum data rate of 5 Gbits/s. [25] There were two possibilities regarding the USB connection. First one was to use a USB 3.0 chip and go through all the protocol and PCB design issues. Second one was to find a module that hid all these issues and allowed an easy USB connection through an interface that is easily implemented in an embedded system. After an extensive search a module produced by Cypress was found with part name EZ-USB FX3. The evaluation board for EZ-USB FX3 can be seen on figure 3.1.
The EZ-USB FX3 module has the following features as stated on Cypress homepage [27]:

- Integration: Full USB 3.0 Peripheral Controller with built-in PHY
- High-Performance: ARM9 with 512 kB RAM for data processing
- Connectivity: I2S, SPI and UART peripherals
- Flexibility: Proprietary 32-bit 100 MHz GPIF™ II
- Low Power: Low 1.2 V core and independent power domains
- Multitasking: 32 configurable endpoints

The GPIF™ II interface makes it easy to get USB 3.0 connectivity in many applications. Cypress has a software suite [28] to configure the EZ-USB FX3. This software suite includes GPIF Designer that simplifies the process of interfacing to other parts of the design. There are also ready interfaces that designers can adapt to their design without having to start from scratch. An example of an interface is a synchronous slave FIFO interface, shown on figure 3.2. As can be seen above a maximum clock speed of 100 MHz is possible on the GPIF II port and to get a maximum USB 3.0 speed a 32 bit parallel port is needed to interface to the GPIF II port. In this design a 16 bit port is sufficient because (786,432,000 bit/s)/(16 bit) = 49.152 MHz. The EZ-USB FX3 module will be interfaced with an FPGA as will be discussed
in next section. The EZ-USB FX3 module will not be configured in this project due to time constraints and awaits future work.

Figure 3.2: GPIF Designer from Cypress is a powerful tool to configure the EZ-USB FX3 module. [29]

3.3 FPGA: Interface the EZ-USB FX3

Field programmable gate arrays (FPGA) are chips that have a matrix of configurable logic blocks (CLB) and I/O blocks as can be seen on figure 3.3. The interconnection between the CLB is programmable and is done with hardware description language (HDL) like VHDL and Verilog. An FPGA can be configured to work as almost any digital circuit. [30] For example FPGA can be programmed to have the same logic function as a complex microcontroller or as simple as an AND gate. When FPGA has been configured the function can be guaranteed to be in real time because it is a combination of hardware logic. The only delays are propagation delays. The multi-channel microphone system has 16 TDM serial audio channels which need to be converted to parallel data, multiplexed and transferred to the EZ-USB FX3 module in real time. This is quite straight forward using FPGA although the learning curve can be quite steep.
On figure 3.4 a block diagram of the intended configuration of the FPGA is shown. This is a simplified diagram and not all signals are shown like the serial clock, frame sync clock and parallel port control signals. First the TDM serial audio data is converted into parallel data. Then it is clocked into a FIFO buffer. There are two FIFO buffers to guarantee no data loss when one buffer is full. The parallel audio data is then clocked to the parallel bus via control module which hides the many modules from the outside world and makes it look like that only one FIFO buffer is present. One frame of data is therefore data from all 128 microphones which is a total of 4096 bits or 512 B of data. This frame is clocked into the EZ-USB FX3 module and sent through the USB 3.0 port.
There are many FPGAs on the market that have plenty of resources available for this task. When selecting an FPGA chip the following requirements where made:

- Have an internal flash and programmed automatically on start-up.
- Easy to use and free development environment.
- Good support and resources for developers.

FPGA from Xilinx, Altera and Lattice Semiconductor were considered. After consideration it was decided that FPGA from Lattice Semiconductors was considered the best choice for this project. The MachXO2 was first considered as a development module but it became out of stock at the time of purchase and thus a MachXO3 development board was chosen instead. This development board has the part number LCMXO3L-6900C-S-EVN and can be seen on figure 3.5. The development software provided by Lattice Semiconductor, Lattice Diamond, is free of charge and easy to use. [31] There good product documentation on Lattice homepage. [32]
3.3.1 FPGA programming

Programming an FPGA is quite different from programming an MCU. When programming an MCU in a programming language like C or C++ it is usually easy to see the software flow because instructions are performed in sequence through an ALU unit. An FPGA is quite different. As was described earlier it is a matrix of CLB and I/O blocks and is configurable on the logic level. Clusters of logic functions can be created to work in parallel, synchronously or asynchronously. One clock signal can be used to drive all the logic clusters or multiple clocks can be used. All this functionality can be described by the popular FPGA programming language VHDL which is a hardware description language. In appendix B.1 a VHDL code is presented. This code describes one TDM SIPO/FIFO module, M0, as shown on figure 3.4. The code is in development and is used to test the functionality of the SIPO/FIFO module.

In Lattice Diamond it is possible to see individual CLB and how they are interconnected. This can sometimes help in debugging a VHDL code. Sometimes errors in the code creates isolated signal paths due to various reasons. This can be quickly spotted when the physical view is turned on. An example of physical view is seen on figures 3.6 and 3.7. Figure 3.6 shows an individual CLB and the signals routed through it. Figure 3.7 on the other hand shows the interconnections between the CLBs used.
Chapter 3. Streaming audio data to PC  3.3. FPGA: Interface the EZ-USB FX3

Figure 3.6: A CLB as shown in Lattice Diamond programming environment for the LCMXO3L-6900C.

Figure 3.7: A physical overview form Lattice Diamond of the interconnections between CLB after a VHDL code has been synthesized and routed.
Chapter 4

Results

In the preceding chapters the design of the multi-channel microphone system has been described. In this chapter the building and testing of the system will be described.

4.1 Building and testing the ADC module

The components where soldered by hand on the ADC module with a normal soldering iron. As a result the components are not as cleanly soldered as would be if a reflow oven were used. On figures 4.1 and 4.2 the PCB boards with components can be seen.

Figure 4.1: Bottom layer showing the vias which connect the ground plains both electrically and thermally.
Chapter 4. Results

4.1. Building and testing the ADC module

After the PCB had been soldered the ATtiny167 MCU was programmed through the Arduino environment using an Arduino Uno board as a programmer. After some testing it was clear that the board was not operating as expected. The CS5368 chip was not being configured through the SPI. After taking a closer look at the datasheet for the CS5368 and the SPI pins it was discovered that the chip select (CS) pin on the CS5368 was grounded instead of being connected to a CS output pin on the MCU. A high to low transition on CS starts a data transfer to the CS5368 and because it was grounded a normal operation of the SPI port was not possible. This was solved by de-soldering the CS pin on CS5368 and connecting it to an I/O pin on the MCU with a low valued resistor as can be seen on figure 4.1. This is a design issue that needs to be changed on future versions of the PCB.

Another design flaw was discovered regarding the buffer amplifier for the $V_Q$ reference voltage from CS5368 to the ADC differential operational amplifier drivers. There is a missing trace from the positive output to the negative input. This is a feedback loop and therefore a proper operation is not possible. This issue will need to be fixed. The THS4521 input drivers have an internal offset voltage of $(V_{S+} - V_{S-})/2$ which is the same as the $V_Q$ reference voltage but with some minor error. The $V_{OCM}$ voltage was measured to be 2.494 V whereas the $V_Q$ voltage was 2.490 V. This is an error of only 0.16% which is considered to be tolerable for the initial testing.

On figures 4.3 and 4.4 the TDM audio stream from CS5368 can be seen. There is nothing connected to input drivers and only random noise is sampled. Getting the
CS5368 to work as expected was not straightforward. The MCU managed to configure the GCTL register correctly through SPI and this was seen by reading back from the register. But the TDM stream was not constantly showing on the oscilloscope and fluctuated on and off. It was found that part of the problem was the MISO pin. When the CS5368 is not in read mode the pin is in high impedance state as shown in CS5368 datasheet. [8] It is suggested to tie this pin to either high or low with a 47 kΩ resistor. This is another issue that needs to be fixed in future versions.

Figure 4.3: The actual TDM audio stream from the CS5368 chip, showing 8 channels. The yellow signal is SCLK, blue FS and green TDM serial stream.
4.1. Building and testing the ADC module

4.1.2 Noise measurements: Power rails

The noise on the power rails will directly affect the performance of the system. As was discussed in section 2.5.1 the power rails are filtered with ceramic bypass capacitor pairs. The 5V power rail for the analog circuits is of special interest regarding noise. The noise was measured both for the 5V analog power rail and the 3.3V digital power rail. Figure 4.4 the noise present on the analog power rail. The peak-peak voltage is 12mV and the rms voltage is 1.4mV with a 20MHz bandwidth limit turned on to use the standard measurements for power supply noise. The oscilloscope has some internal noise which adds to the noise measurements. Attempts will not be made to account for that in the measurements. When the probes are shorted to ground approximately 1mVrms noise was present. The noise measurements for the power rails can be seen in table 4.1.

Figure 4.4: The actual TDM audio stream from the CS5368 chip, showing 1 channel. The yellow signal is SCLK, blue FS and green TDM serial stream.
Chapter 4. Results

4.1. Building and testing the ADC module

20MHz bandwidth

<table>
<thead>
<tr>
<th>Source</th>
<th>Noise (peak-peak)</th>
<th>Noise (rms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oscilloscope noise</td>
<td>6mV pp / 1mV rms</td>
<td></td>
</tr>
<tr>
<td>Analog supply</td>
<td>12mV pp / 1.4mV rms</td>
<td></td>
</tr>
<tr>
<td>Analog supply, digital ground</td>
<td>25mV pp / 4.7mV rms</td>
<td></td>
</tr>
<tr>
<td>Digital supply</td>
<td>33mV pp / 5mV rms</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Power supply noise measurements.

The results in table 4.1 are considered to be quite good. It is interesting to compare these numbers to a high quality power supply from Keysight, E3631A, which has a noise of 2mV pp / 350\(\mu\)V rms. [34] Although it would have been desirable to further reduce the noise on the analog power rail a noise of 12mV pp / 1.4mV rms for this initial design of the ADC module the noise level reached is considered acceptable.

It is clear that the measures taken to reduce noise in the analog part as discussed in sections 2.5.1 and 2.5.2 are effective as can be seen from the fact that the noise level is higher in the digital part than the analog part. It is also interesting to see the noise level when the analog noise is measured with reference to the digital ground as seen in table 4.1 and figure 4.7. It is obvious that when the return current of the signal entering the oscilloscope has to route through the noisy digital ground the noise level is much higher, 25mV pp / 4.7mV rms compared to 12mV pp / 1.4mV rms. This is of great interest and emphasizes the importance of a good ground plane layout of mixed-signal PCBs.

Figure 4.5: Noise on the 5V analog power rail.
Figure 4.6: Noise on the 3.3V digital power rail.

Figure 4.7: Noise on the 5V analog power rail when digital ground is used.
4.1.3 Noise measurements: Input drivers

The output noise measurements of the THS4521 input drivers can be seen in table 4.2 and on figure 4.8. When no input signal is present the inputs and outputs stabilize at $V_{OCM}$, in this case 2.50V when measured with a multimeter. A 20MHz bandwidth limit was used when differential outputs where measured with an oscilloscope. It is not possible to test the rated performance specifications in the datasheet of operational amplifiers due to other noise sources. Connecting both input pins together to ground or supply voltage did not impact the output noise. The purple signal on figure 4.8 is the difference between the two outputs of the ADC driver. There is not much difference between the noise on the two outputs which means that it is mostly a common mode noise. This noise should therefore cancel out to large extent in the CS5368.

$$\begin{align*}
V_{out+} & : 25mV_{pp} \backslash 5.0mV_{rms} \\
V_{out-} & : 24mV_{pp} \backslash 5.1mV_{rms} \\
V_{out+} - V_{out-} & : 10.7mV_{pp} \backslash 1.8mV_{rms}
\end{align*}$$

Table 4.2: ADC driver output noise measurements.

Figure 4.8: Noise at the output of one ADC driver. The green and yellow are differential outputs and the purple one is the difference between yellow and green.
4.1.4 Heat dissipation

The circuit was tested with a master clock frequency of 15 MHz, 25 MHz and 36 MHz and the heat dissipated measured with an infra-red thermometer. The accuracy is not expected to be high but this measurement gives an idea of the heat dissipation. In table 4.3 is the temperature of CS5368 as a function of frequency. A 50MHz clock source was not readily available and therefore the temperature of CS5368 was estimated to be 86°C with 1.8°C/MHz from 36 MHz. This temperature is considerable higher than anticipated due to the fact that in the datasheet it is stated that the power consumption is less than 680 mW and would not dissipate a lot of heat. In section 2.1.9 it was stated that the datasheet for CS5368 was not clear on how power consumption changes proportionally with master clock frequency.

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Temperature</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 MHz</td>
<td>50°C</td>
<td></td>
</tr>
<tr>
<td>25 MHz</td>
<td>53°C</td>
<td>0.3°C/MHz</td>
</tr>
<tr>
<td>36 MHz</td>
<td>61°C</td>
<td>1.4°C/MHz</td>
</tr>
<tr>
<td>50 MHz</td>
<td>86°C</td>
<td>Estimated with 1.8°C/MHz</td>
</tr>
</tbody>
</table>

Table 4.3: CS5368 heat dissipation.

The LP38693, 3.3 V, voltage regulator was measured to be 46°C at a master clock of 36 MHz. This is well below the allowed maximum temperature of 90°C. The temperature is not expected to rise considerably due to higher clock frequency.

4.2 Testing the FPGA module

In figure 4.9 test setup of the current state of the project can be seen. The ADC module is connected to the FPGA running the test program in appendix B.1. The test module sipo_tdm_buff should make the TDM serial stream into a 16 bit parallel stream and store in a FIFO buffer which has the depth of 16. One TDM frame is thus stored in the FIFO buffer. When the buffer is full it signals it with a data read signal (dr). The serial clock (SCLK) from CS5368 is connected to sclk and pclk input signals, FS to frame_s and TDM to serin. An oscilloscope is used to monitor the relevant signals. Figure 4.10 shows the above operation. The dr signal goes high on the rising edge of FS which is a correct operation. The serial output is taken from the FPGA through serout. This was done to make sure that the TDM serial data was correctly transferred into the shift registers. During the test the rd (read) signal was held high so the FIFO buffers would be clocked out to the parallel bus. As soon as the FIFO buffers are empty the dr signal should go low.
Figure 4.9: The ADC module connected to the FPGA. The FPGA is programmed with the test code in appendix B.1.

Figure 4.10: The TDM data stream is clocked into the FPGA and data ready (dr) signal goes high on the rising edge of each frame sync pulse (FS).
4.2.1 MachXO3L problems

4.3 USB module: EZ-USB FX3

As was stated in section 3.2 the EZ-USB FX3 module was not configured because of time constraints.
Chapter 5

Conclusion

In the preceding chapters the design of a multi-channel microphone system has been described. Designing a system consisting of different modules that have to work together can be challenging. Many problems had to be overcome. Some of the biggest problems had to do with decision making. It was not straight forward to decide how to build up the system. There are many solutions available and an extensive search was needed to find the solutions that fit the design. The following goals were set in section 1.2:

1. Make decision on the fundamental design of the system.
2. Decide how the data is to be transferred through USB.
3. Design the ADC module.
4. Get at least one ADC module to work as expected.
5. Stream audio data through USB 3.0.

All of these goals were reached, at least to some extent. A decision was made about the fundamental design of the system. Three design topologies were considered and one of them chosen. This design topology included an FPGA and a separate USB 3.0 module and therefore the second goal was reached, to decide how to transfer data through USB. Taking these decisions was a lengthy process. But the outcome is considered to be good. The EZ-USB FX3 module from Cypress is of great interest. This module makes USB 3.0 connection possible for almost all kinds of embedded systems. It was unfortunate that it was not possible to program this module due to time constraints. The FPGA chip included in the data stream solution is also of great interest. There was no prior knowledge of programming FPGAs and therefore an in-depth study of FPGA programming was needed. After this study it was more obvious that an FPGA in this design was the right choice. FPGA are configured on the logic
level and when a high number of data streams are involved an FPGA is definitely a choice to consider.

The design of the ADC module was a lengthy process. Every component needed to be carefully evaluated and chosen while the schematic was created. All components packages and footprints on the PCB needed to match. The components also needed to be chosen according to their application. For example some capacitors needed to be C0G rated for stability reasons. Goal number 4 was almost reached. It was possible to get TDM stream from the ADC module. There were some problems with the communication between the MCU and CS5368 but with some trial and error it was possible to get a correct operation.

5.1 Future work

The multi-channel microphone system has a lot of potential to become a fully functional system capable of sampling data from up to 128 microphones and even more. For that to become a reality it is necessary to continue working on the system.

5.1.1 ADC module

The PCB board needs some redesigning. The design flaws mentioned in section 4.1.1 need to be fixed. Also more needs to be done to reduce noise. That can be done with better ground place layout and bypass capacitor network. The communication between the MCU and CS5368 needs some clarification and also to add pull down or pull up resistors were needed to prevent floating inputs. It is also worth mentioning that the MCU can be put in a sleeping mode after programming the CS5368 to reduce noise.

5.1.2 FPGA module

The program for the FPGA module needs to be finished. The SIPO/FIFO operation is working but more tests are needed. The multiplexing of the TDM stream into the USB module needs to be finished. The FPGA modules used stoped working correctly. The FTDI chip which is a USB-Serial bridge, stopped working. It is also possible to program the MachXO3L with a MCU. Lattice Semiconductor provides a C code for embedded programming of the FPGA. This code needs some modification to work on specific MCU. Attempts were made to program a Pic32 MCU to act as a programmer for the FPGA but due to limited time it was not possible to finish.
5.1.3 EZ-USB FX3 module

The EZ-USB FX3 needs to be programmed. Cypress provides excellent design tool and documentation along with an example code which makes the programming easier. This module is considered to be a good solution to stream the data through USB 3.0. The need for USB 3.0 specific knowledge is almost totally eliminated. It is highly recommended to use this module in this project.
Bibliography


Appendix A

ADC module

A.1 Schematic diagrams

In this appendix the schematic diagram of the ADC module will be presented.
Figure A.1: Schematic diagram of the CS5368 chip and how it is connected.
Figure A.2: Schematic diagram of the ADC input driver.
Figure A.3: Schematic diagram of all ADC input drivers.
Figure A.4: Schematic diagram of $V_Q$ buffer and bypassing capacitor network.
Figure A.5: Schematic diagram of ATtiny167 MCU and header pins connections.
Figure A.6: Schematic diagram of the voltage regulators for analog and digital part.
A.2 PCB layout

Figure A.7: The PCB layout showing all layers.
Figure A.8: The PCB layout showing only top layer.
Figure A.9: The PCB layout showing only layer 2.
Figure A.10: The PCB layout showing only layer 3.
Figure A.11: The PCB layout showing only bottom layer.
Appendix B

FPGA

B.1 VHDL test program for TDM SIPO/FIFO module

The VHDL program in this section was used to test the TDM SIPO/FIFO module for correct operation. This program is not intended for the end product.

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use IEEE.NUMERIC_STD.ALL;
library machxo3;
use machxo3.all;

entity sipo_tdm_buff is
    generic(
        FRAME_SIZE : integer := 256; --Bits
        DATA_W : integer := 16; --Bits
        BUFFER_DEPTH : integer := 16; --FRAME_SIZE/DATA_W: Number of DATA_W in buffer
        DATA_ADDR_W : integer := 4; --log2(BUFFER_DEPTH)
        DATA_ADDR_MAX : integer := 15; --BUFFER_DEPTH-1
        BUFFER_CNT : integer := 2; --This buffer holds one TDM frame.
        BUFFER_ADDR_W : integer := 1; --log2(BUFFER_CNT)
        BIT_CNT_W : integer := 5
    );
    port(
        sclk : in std_logic;
        pclk : in std_logic;
        frame_s : in std_logic;
    );
```


Appendix B. FPGA

B.1. VHDL test program for TDM SIPO/FIFO module

serin : in std_logic;
n_reset : in std_logic;
dr : out std_logic; --Data ready, tdm buffer
err : out std_logic;
rd : in std_logic; --Read parallel data, makes data ready on parout
--Keep high while reading data.
ovfl : out std_logic;--Overflow bit. If Buffer is not read before it
--is overwritten, this bit is put high. Cleared
--is next successful buffer read.
parout : out std_logic_vector(DATA_W-1 downto 0);
serout : out std_logic;
parout_2 : out std_logic_vector(DATA_W-1 downto 0);
check_out : out std_logic);
end sipo_tdm_buff;

architecture behavioral of sipo_tdm_buff is

--Buffers for TDM frame.
type reg_file_type is array (0 to (BUFFER_DEPTH-1)) of
    std_logic_vector(DATA_W-1 downto 0);
--type reg_file_array_type is array (0 to BUFFER_CNT-1) of reg_file;
signal sig_serin : std_logic;
signal sig_sclk : std_logic;
signal sig_pclk : std_logic;
--Shifting register for parallel data.
signal sig_shift_reg : std_logic_vector(DATA_W-1 downto 0);
signal sig_shift_finish : std_logic := '0';
signal sig_shift_count : integer := 0;
signal sig_parout_buff : std_logic_vector(DATA_W-1 downto 0)
    := (others => 'Z');
--Has frame sync occured
signal sig_frame_s : std_logic := '1';
signal sig_tdm_frame_buff_0 : reg_file_type;
signal sig_tdm_frame_buff_1 : reg_file_type;
signal sig_buff_ready_flag : std_logic_vector(0 to BUFFER_CNT-1)
    := (others => '0');
Appendix B. FPGA  B.1. VHDL test program for TDM SIPO/FIFO module

signal sig_buff_rd_finish_flag : std_logic_vector(0 to BUFFER_CNT-1) := (others => '0');
signal sig_buff_read_finish : std_logic := '0';
signal sig_buff_ready : std_logic := '0';
signal sig_wr_ptr : integer range 0 to DATA_ADDR_MAX := 0;
signal sig_wr_ptr_nxt : integer range 0 to DATA_ADDR_MAX := 0;
signal sig_rd_ptr : integer range 0 to DATA_ADDR_MAX := 0;
signal sig_rd_ptr_nxt : integer range 0 to DATA_ADDR_MAX := 0;
--Tells what tdm buffer is being written to.
signal sig_buff_active_ptr : integer range 0 to BUFFER_CNT-1 := BUFFER_CNT-1;
--Tells what tdm buffer is ready.
signal sig_buff_ready_ptr : integer range 0 to BUFFER_CNT-1 := BUFFER_CNT-1;
signal sig_ovfl : std_logic := '0';
signal sig_dr : std_logic := '0';
signal sig_err_wr : std_logic := '0';
signal sig_err_rd : std_logic := '0';
signal sig_buffer_read_finish : std_logic := '0';
signal sig_rd_finish : std_logic := '1';
signal sig_check_out : std_logic;
begin
  sig_serin <= serin;
  sig_sclk <= sclk;
  sig_pclk <= pclk;
  check_out <= sig_check_out;
  --Process to control sig_frame_s bit.
  --This signals if frame sync pulse has occured
  process(frame_s, n_reset)
  begin
    if(n_reset = '1') then
      if(rising_edge(frame_s)) then
        sig_frame_s <= '1';
      end if;
    else
      sig_frame_s <= '0';
    end if;
  end process;
end

74
end process;

--Process to do serial to parallel conversion
process (sig_sclk)
    variable shift_reg_var : std_logic_vector(DATA_W-1 downto 0);
    variable sig_shift_count_var : integer;
    variable sig_wr_ptr_var : integer;
    variable sig_buff_ready_ptr_var : integer;
    variable sig_buff_active_ptr_var : integer;
    variable sig_buff_ready_flag_var
        : std_logic_vector(0 to BUFFER_CNT-1);
    variable sig_buff_rd_finish_flag_var
        : std_logic_vector(0 to BUFFER_CNT-1);
begin
    if(rising_edge(sig_sclk)) then
        if(sig_rd_finish = '1') then
            dr <= '0';
        end if
        if(n_reset = '1' and sig_frame_s = '1') then
            sig_shift_count_var := sig_shift_count;
            sig_wr_ptr_var := sig_wr_ptr;
            sig_buff_ready_ptr_var := sig_buff_ready_ptr;
            sig_buff_active_ptr_var := sig_buff_active_ptr;
            sig_buff_ready_flag_var := sig_buff_ready_flag;
            sig_buff_rd_finish_flag_var := sig_buff_rd_finish_flag;
            shift_reg_var(DATA_W-1 downto 0)
                := sig_shift_reg(DATA_W-1 downto 1);
            shift_reg_var(DATA_W-1) := sig_serin;
            serout <= sig_serin;
            sig_shift_count_var := sig_shift_count_var + 1;
            if(sig_shift_count_var < DATA_W) then
                sig_shift_finish <= '0'; --Reset shift_finish
            else
                sig_check_out <= not sig_check_out;
                sig_shift_finish <= '1';
                sig_shift_count_var := 0;
            end if;
        end if;
    end if;
end process;
case sig_buff_active_ptr_var is
  when 0 =>
    sig_tdm_frame_buff_0(sig_wr_ptr_var) <= shift_reg_var;
    parout_2 <= sig_tdm_frame_buff_0(sig_wr_ptr_var);
    sig_err_wr <= '0';
  when 1 =>
    sig_tdm_frame_buff_1(sig_wr_ptr_var) <= shift_reg_var;
    parout_2 <= sig_tdm_frame_buff_1(sig_wr_ptr_var);
    sig_err_wr <= '0';
  when others =>
    sig_err_wr <= '1';
end case;

sig_wr_ptr_var := sig_wr_ptr_var + 1;

if (sig_wr_ptr_var = BUFFERDEPTH) then
  dr <= '1';
  if (sig_buff_rd_finish_flag_var(sig_buff_ready_ptr_var) = '1') then
    sig_buff_ready_flag_var(sig_buff_ready_ptr_var) := '0';
  end if;
  sig_wr_ptr_var := 0;
  sig_buff_active_ptr_var := sig_buff_active_ptr_var;
  sig_buff_active_ptr_var := sig_buff_active_ptr_var + 1;
  -- Check if active ptr is more than max ptr, and zero
  if (sig_buff_active_ptr_var = BUFFER_CNT) then
    sig_buff_active_ptr_var := 0;
  end if;
  if (sig_buff_ready_flag_var(sig_buff_ready_ptr_var) = '1') then
  -- Read not finished from ready buffer before writing to it, 
  -- set overflow bit.
    ovfl <= '1';
  else
    ovfl <= '0';
  end if;
  sig_buff_ready_flag_var(sig_buff_ready_ptr_var) := '1';
  -- sig_buff_ready <= '1';
  end if;
end if;
else
    --Reset
    sig_shift_count_var := 0;
    sig_shift_finish <= '0';
    sig_buff_active_ptr_var := 0;
    sig_wr_ptr_var := 0;
    sig_buff_ready_flag_var := (others => '0');
end if;
end if;

--Update signals
sig_shift_reg <= shift_reg_var;
sig_shift_count <= sig_shift_count_var;
sig_wr_ptr <= sig_wr_ptr_var;
sig_buff_ready_ptr <= sig_buff_ready_ptr_var;
sig_buff_active_ptr <= sig_buff_active_ptr_var;
sig_buff_ready_flag <= sig_buff_ready_flag_var;
end process;

process(sig_pclk, n_reset)
variable sig_rd_ptr_var : integer;
variable sig_buff_rd_finish_flag_var : std_logic_vector(0 to BUFFER_CNT-1);
variable parout_var : std_logic_vector(DATA_W-1 downto 0);
begin
    if(rising_edge(sig_pclk)) then
        sig_rd_finish <= '0';
        if(n_reset = '1' and sig_frame_s = '1') then
            sig_rd_ptr_var := sig_rd_ptr;
            sig_buff_rd_finish_flag_var := sig_buff_rd_finish_flag;
            if(rd = '0') then
                parout_var := (others => 'Z');
            else
                if(sig_buff_ready_flag(sig_buff_ready_ptr) = '1') then
                    case sig_buff_ready_ptr is
                    when 0 =>
                        parout_var := sig_tdm_frame_buff_0(sig_rd_ptr_var);
                        sig_err_rd <= '0';
                    end case;
                end if;
            end if;
        end if;
    end if;
end process;
when 1 =>
    parout_var := sig_tdm_frame_buff_1(sig_rd_ptr_var);
    sig_err_rd <= '0';
when others =>
    sig_err_rd <= '1';
end case;
sig_rd_ptr_var := sig_rd_ptr_var + 1;

if(sig_rd_ptr_var < BUFFERDEPTH) then
    sig_buff_rd_finish_flag_var(sig_buff_ready_ptr) := '0';
else
    sig_rd_ptr_var := 0;
    sig_buff_rd_finish_flag_var(sig_buff_ready_ptr) := '1';
    sig_rd_finish <= '1';
end if;
else
    sig_err_rd <= '1';
end if;
else
    sig_rd_ptr_var := 0;
    sig_buff_rd_finish_flag_var := (others => '0');
end if;
end if;

parout <= parout_var;
sig_buff_rd_finish_flag <= sig_buff_rd_finish_flag_var;
sig_rd_ptr <= sig_rd_ptr_var;
end process;
end behavioral;