
DAT096

**Ultrasonic Sound Localization using
Microphone Arrays**

**Product Documentation
Group M3**

Ashwin Kumar Balakrishnan
Hezhe Xiao
John Croft
Prema Manickavasagam

May 29, 2020



CHALMERS
UNIVERSITY OF TECHNOLOGY

1 The system design

The main aim of this system design is to locate an ultrasonic acoustic source, which uses a 24 MEMS microphone array and a beamforming algorithm implemented on a FPGA to perform the directional source location.

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

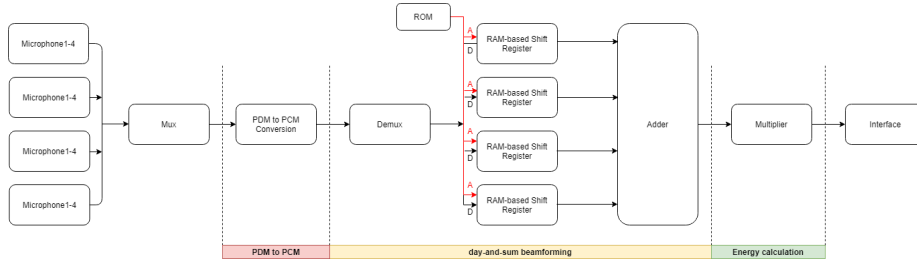


Figure 1: System architecture diagram

2 Time multiplexer and de-multiplexer

A time multiplexer is used to access the multiple channel mode of the FIP/CIC filters to use more than one microphone. It sends inputs through the AHB bus of the filters on every rising edge of the clock. The input from each channel is sent on each rising edge and after all the data is sent **Tlast** is set to 1. The number of channels in the IP blocks of the CIC/FIR filters is changed according to the number of mics used and the clock frequency is adjusted accordingly. After conversion, a de-multiplexer is used which splits the multiplexed signals into original signals based on the clock.

TESTS :- First we used an audible source as input to the microphones and listened to the output and therefore verified that it works. Later we changed the source to be ultrasonic and verified the functionality by plotting the spectrum of the output signal in AUDACITY.

3 PDM to PCM conversion

The system design implements the conversion from 1-bit PDM to PCM using one cascaded integrator-comb decimating filter(CIC filter), one half-band FIR filter, one low-pass FIR filter and one high-pass filter to output the 16-bit filtered data at a 100 kHz rate of PCM sampling frequency.

After conversion, the interleaved data is separated using a demultiplexer (demux) and a bitstream containing the number of clock cycles corresponding to the delay values for each scanning angle is driven to the input address of the shift register.

3.1 Delay value calculation

The delay values are calculated in the 'delay_compute.m' MATLAB script file attached. According the equations below, the microphone array given, has a fixed distance between can't be changed. The parameters that could be changed are the number of microphones N and the sampling of PDM f_{clk} (3.2 Mhz). The effect of source frequency and aliasing can be found by the beampattern MATLAB script file.

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

$$N_{clk} = ceil(Delay * f_{clk})$$

After getting the delay values for four microphones, transferring it to hexadecimal data save it as newvalue.coe file and upload it to ROMs, the configure of each ROM is 16 bits for width and 30 number for depth.

3.2 Clocking for PDM to PCM conversion

In order to increase the number of microphones, the clock frequency has to be increased as well. The clock frequency of the mics should be changed according to the number of microphones and PDM sampling frequency for each microphone. Therefore, the clock frequency of 12.8 Mhz(4*3.2 Mhz).

3.3 Filters in PDM to PCM conversion

In order to use ultrasonic frequencies, the filter co-efficients and clocking need to be change to meet the requirement 3.2 Mhz of PDM sampling frequency. The filter co-efficients are generated in MATLAB using the filterDesigner tool and stored in a halfband_ultrasonic.coe and lowpass_ultrasonic.coe file. The cutoff and the sampling frequency is decided based upon the source frequency.

4 RAM based shift register and Block memory generator

The four signal data from Demux block are propagated out of shift registers at each angle by the delay values saved in ROMs. There are totally 120 set of clock cycles, from 30 degree to 150 degree corresponding to each microphone in newvalue.coe file. The values are accessed by the memory address and then outputted to the shift register. The delayed signals are then sent to the adder.

5 Finite state machine of scan block

In order to control the shift register and energy calculation block, we need to use finite state machine.

The IDLE state is to wait for the switch on the FPGA turns on and all the data is finished to demux. The delay state is used for all the four microphones propagated out of the shift registers at each degree and the sum state is to add the amplitude together. Every time the system changes to the next four address of next angle, the number of delay clock cycle is already saved in ROMs, it will return to delay value to assert a counter and read the new delay from ROMs

We use a flag signal of scan_finish to indicate that all delay values are implemented for four microphones and another flag signal of adder_enable to control the energy calculation block. The figure showed below illustrates how the micros are propagated out of the shift registers and energy calculation block is controlled by scanning block.

6 Energy calculation

The arithm_block is to use for calculation the delayed signal from shift registers, which squares the input signal and the squared signals are compared to get the one with the greatest energy in order to identify the location of the sound source. The particular implementation used in this project attempts to find a local maximum in the squared signal.

There are other methods for estimating the energy, such as time-integrating the signal over a fixed period, but due to issues with the input and output there was no opportunity to test the viability, advantages and drawbacks of these different methods.

7 Data storage and communication (to PC)

The design in vivado is synthesised and the bitstream is downloaded to the hardware. The recorded data or energy data is then accessed by a PC via ethernet using GRMON. The sampled data is imported to Audacity and then verified using a spectrum analyser. Energy data is imported in Matlab to analyze the direction of sound location.