A Novel Approach towards Sound Source Localization

A Project Report

submitted by

SIDDHARTH SHEKAR

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY
(ELECTRICAL ENGINEERING)

AND

MASTER OF TECHNOLOGY
(MICROELECTRONICS AND VLSI DESIGN)

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

MAY 2013

THESIS CERTIFICATE

This is to certify that the thesis titled **A Novel Approach towards Sound Source Localization**, submitted by **Siddharth Shekar**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology and Master of Technology**, is a bona fide record of the research work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Nitin Chandrachoodan
Research Guide
Associate Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 28^{th} May 2013

ACKNOWLEDGEMENTS

I would like to thank Dr. Nitin Chandrachoodan for giving me the freedom to pursue my ideas while still ensuring that I was on the right track. Indeed, if he hadn't let me "fly by the seat of my pants", the project would have turned out very differently.

I am deeply grateful towards my lab-mates Abishek, Bharath, Bhargava, Numaan, Prasad, Sudharshan and Vignesh for hearing out all (and there were many of those) my cribs during the course of the project. Bouncing my ideas off them led to many revelations which I certainly wouldn't have found on my own.

Thanks are due to my wing-mates Midhun, Aditya, Advaith, Harsha, Kaushik, Vishruth and Lohit without whom my stay at IITM wouldn't have been half as enjoyable an experience as it was. Late-night discussions on things relevant and irrelevant are something that I will cherish for the rest of my life.

Finally, none of this would have been possible without the support and encouragement that my parents and sister have given me. Their unwavering faith in me has been a source of great motivation and continues to remain so.

ABSTRACT

KEYWORDS: Source Localization; Hard-limiting; Time difference of arrival; Delay and sum beamforming; FPGA; Time to digital converter.

Source localization has been a topic of active research interest for several decades now. Microphone arrays form a core component of the hardware involved. Regular as well as non-regular arrangements of microphones have been explored in great depth. Most localization algorithms rely on estimating the time delay of arrival of the input signal between pairs of microphones. A commonly used technique for doing so is the Generalized Cross-Correlation (GCC) method. The issue with cross-correlation, however, is the high computational complexity, which is a function of the frame length under consideration.

In this thesis, a novel approach towards estimating the time delay of the signal received at pairs of microphones is presented. Since the time delay is the information to be extracted, we threshold the signal so that only information about the sign bit is retained. The delay between corresponding edges of these binary signals is then estimated using a Time to Digital Converter (TDC).

Two methods for source localization are explored - the Time Difference of Arrival (TDOA) method, which is a hard assignment and the Delay and Sum BeamForming (DSBF) method, which is a soft assignment. We also propose a simplified computation scheme for the DSBF approach which enables us to achieve high-resolution real-time source localization.

Simulation results for up to 9 sensors and actual implementation results using 4 sensors show that the new approach matches the old approach with reasonable accuracy. Real-time visualization is achieved with both TDOA and DSBF approaches.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS i

ABSTRACT ii

LIST OF TABLES vi

LIST OF FIGURES vii

ABBREVIATIONS viii

NOTATION ix

1 Introduction 1

1.1 Microphone arrays and related applications 1

1.2 Existing approaches 2

1.3 Time to Digital Converters 3

1.4 Contribution of this work 4

1.5 Structure of the thesis 5

2 Hardware Details 6

2.1 Description of the front-end 6

2.2 Time delay estimation using FPGA 7

2.3 Verilog Implementation 9

2.3.1 delayTDC 10

3 TDOA 12

3.1 Implementation Details 13

3.1.1 Spatial aliasing 13

3.1.2 Reduction in sampling rate 13

3.1.3 Conversion to spherical coordinates 14

3.2 Verilog implementation 15
3.2.1 Top 15
3.2.2 TDOALocalizer 16
3.2.3 DCM 16
3.2.4 tdoaSpherical 16
3.2.5 vgaHSVS 16
3.2.6 generateRGB 17

4 DSBF **18**
4.1 Implementation Details 19
4.2 Verilog Implementation 21
4.2.1 dsbfCalculator 22
4.2.2 coordinateGenerator 23
4.2.3 CORDIC 23
4.2.4 intensityCalculator 23
4.2.5 similarityCalculator 24
4.2.6 generateRGB 24

5 Results and Discussion **25**
5.1 TDOA 25
5.1.1 Simulation Results 25
5.1.2 Implementation Results 26
5.2 DSBF 27
5.2.1 Simulation Results 27
5.2.2 Implementation Results 29

6 Conclusions and Scope for Future Work **31**
6.1 Contribution of the thesis 31
6.2 Scope for Future Work 32

A Derivation of source coordinates using delay values **33**

B Efficiently computing the sum of 4 numbers **35**

C Noise Analysis **37**

C.1 Random noise 37

C.2 Quantization Noise 38

D Sensor position calibration **40**

References **44**

LIST OF TABLES

5.1 Resource utilization for the TDOA algorithm 26
5.2 Resource utilization for the DSBF algorithm 30

LIST OF FIGURES

1.1 Discarding amplitude information 4

2.1 Receiver Front-end 7
2.2 Photograph of the circuit on a breadboard 8
2.3 Time delay estimation 8
2.4 4-phase shifted clock signal scheme to increase TDC resolution . . . 9
2.5 Delay TDC Architecture 10

3.1 Sensor array 12
3.2 Architecture level design 15

4.1 DSBF Principle 18
4.2 Typical received signals 20
4.3 Example of summing 4 signals 21
4.4 Architecture for implementing the DSBF algorithm 22

5.1 Simulation results for the spherical TDOA approach 26
5.2 TDOA output as seen on a VGA display 27
5.3 Simulation results for DSBF using 4 receivers and 1 transmitter . . . 28
5.4 Simulation results for DSBF using 9 receivers and 1 transmitter . . . 28
5.5 Simulation results for DSBF using 4 receivers and 2 transmitters . . 29
5.6 Simulation results for DSBF using 9 receivers and 2 transmitters . . 29
5.7 DSBF result as seen on a VGA display 30

C.1 Quantization error at $z = 2$m 39
C.2 Quantization error at $z = 2$m assuming spherical localization 39

D.1 Unknown sensor locations 40

ABBREVIATIONS

TDOA	Time Delay/Difference of Arrival
GCC	Generalized Cross-Correlation
WSS	Wide Sense Stationary
MUSIC	MUltiple SIgnal Classification
ML	Maximum Likelihood
Tx	Transmitter
Rx	Receiver
FPGA	Field Programmable Gate Array
ASIC	Application Specific Integrated Circuit
ADC	Analog to Digital Converter
DCM	Digital Clock Manager
IP	Intellectual Property
MSB	Most Significant Bit(s)
LSB	Least Significant Bit(s)
TDC	Time to Digital Converter
LED	Light Emitting Diode
VGA	Video Graphics Array
DSBF	Delay and Sum BeamForming
CORDIC	Co-Ordinate Rotation DIgital Computer
LUT	Look-Up Table
DSP	Digital Signal Processing

NOTATION

ϕ	Phase lag/Azimuthal angle
A	Amplitude of received signal
N_{mic}	Number of microphones
α	Inter-sensor distance along a side of the array
Δ_{ij}	Arrival time difference for the signal at sensors i and j
τ	Time for the signal to reach sensor at origin
c	Speed of sound in air
f_{signal}	Frequency of signal incident on the array
$\Delta_{c,i}(P)$	Calculated Δ_{i0} assuming a source at P
$\Delta_{o,i}(P)$	Observed Δ_{i0} assuming a source at P
$s_i[k]$	Samples of received signal at sensor i at index k
$I(P)$	Intensity of the DSBF signal assuming a source at P
$\delta_i(P)$	Difference between $\Delta_{c,i}$ and $\Delta_{o,i}$
σ_i^2	Standard deviation of the jitter in the signal at sensor i
SNR_i	Signal-to-noise ratio at sensor i

CHAPTER 1

Introduction

1.1 Microphone arrays and related applications

Microphone arrays have been used for sound source localization since the early 90's (Brandstein and Silverman, 1997). Possibly the greatest advantage that these systems have is the ability to "steer"(Kellermann, 1991) so that a particular point in space is focused. Sound emanating from that point is consequently enhanced whereas that from other sources is relatively attenuated. Further, it is a known fact that having several copies of the same signal can be used to improve the SNR of the signal. An array of microphones can thus be used to obtain performance levels at par with directed, high-performance microphones with the added benefit of electronically being able to control the directionality. Several commercial products have also been launched based on the idea.

Several applications for the microphone array have already been explored(Brandstein and Silverman, 1997). These include:

- Teleconferencing (Khalil *et al.*, 1994; Kellermann, 1991; Addeo *et al.*, 1994)
- Speech recognition (Moore and McCowan, 2003; Kiyohara *et al.*, 1997)
- Speaker identification (Lin *et al.*, 1994; Busso *et al.*, 2005)
- Speech acquisition in noisy environments (Fischer and Simmer, 1996)
- Underwater sensing (Yan *et al.*, 2012)
- Robotics (Kwon *et al.*, 2008)

An important requirement of several of these applications is the accurate localization of the source. For audio-based applications, knowledge of the presence of coherent

noise sources may be necessary. Microphone arrays can help in locating a particular source and tracking its movement. The source location information can be extended to be used by cameras as well in order to track the source movement without any human involvement.

Reliable tracking of sources, however, places emphasis on real-time or near real-time performance of the algorithms being used. Thus, the algorithm needs to be robust as well as computationally light so that the source is tracked accurately and continuously.

1.2 Existing approaches

Source localization strategies can be classified into three broad categories

Subspace based techniques

This class of techniques relies on localizing source(s) based on the particular nature of the eigenvalues of the correlation matrix. Since the correlation matrix is rarely known *a priori*, in most cases, this is computed assuming that the received signals and the corrupting noise are both Wide Sense Stationary (WSS). The popular MUltiple SIgnal Classification (MUSIC) (Schmidt, 1986) and all of its variants (Mosher and Leahy, 1999; Stoica *et al.*, 1995) and others (Ziskind and Wax, 1988) belong to this class of algorithms. Although these methods can yield high resolution localization, the computational complexity is extremely high. Further the ensemble averaging that is required to obtain the data matrix makes tracking difficult because it assumes physical stationarity of the source.

Beamforming based techniques

The optimal Maximum Likelihood (ML) estimator in this type of localization amounts to finding the peak power for the received signal by scanning the entire search space through electronic steering of the beam formed by the receivers (Chen *et al.*, 2002). Relatively simple beamforming schemes merely alter the delay values, however several more robust and complicated schemes exist, which consider change in amplitude values as well (Lorenz and Boyd, 2005). The sec-

ond step usually involves iteratively finding the peak of the output power, which is hard, in general, since beamforming techniques generally produce local maximas as well (Brandstein and Silverman, 1997). These techniques too suffer from high computational complexity requirements and consequently do not offer optimal real-time results.

TDOA based techniques

This class of techniques consist of 2 stages - first, to find the relative delays between the received signals at different sensors and second, to use this information to find the location of the source. In other words, the delay values and the relative sensor positions are used to generate possible loci for the source and the intersection of these loci is optimized in some sense (Valin *et al.*, 2003). However, techniques have also been developed to try and combine these 2 steps into a single one (Rui and Florencio, 2003). These techniques are possibly the least computationally expensive. However, they suffer from a major disadvantage. The intrinsic assumption in these techniques is that there is a single source to be localized. Presence of multiple coherent sources results in unreliable results.

Of late, there has been active interest in trying to implement these techniques on FPGAs (Zimmermann and Studer, 2010; Lédeczi *et al.*, 2005; Nguyen *et al.*, 2003; Jin *et al.*, 2008). The amount and flexibility of the resources available on FPGAs make them ideal candidates for achieving real-time performance, as is demonstrated by (Zimmermann and Studer, 2010).

1.3 Time to Digital Converters

As the name suggests, TDCs convert a time interval into a corresponding digital value. They are used in a variety of areas but are possibly best known for their use in high energy physics experiments where extremely fine time resolution is necessary. Indeed, sub-nanosecond resolution has been reported (Jansson *et al.*, 2006; Dudek *et al.*, 2000; Kalisz *et al.*, 1997). Tapped delay lines, Vernier delay lines and interpolators are generally employed in order to achieve this level of resolution (Kalisz, 2004).

FPGAs are used extensively in this area because of their ability to create long delay lines, which allow for fast prototyping even though Application Specific Integrated Circuits (ASICs) might offer better performance.

1.4 Contribution of this work

In this work, we use a TDC to estimate delays and consequently use this data in the localization scheme. We note that for closely spaced sensors, the amplitude of the received signal will be approximately the same if only one source is present. Hence, the information pertaining to the delay is captured more by the phase difference between the received signals. Thus, it should be safe to discard the amplitude information altogether and consider merely the sign of the received signal as shown in Figure 1.1. This is achieved by the use of a comparator directly at the output of each receiver. Comparing the received signal to the ground level directly gives us the sign bit in the form of a binary signal.

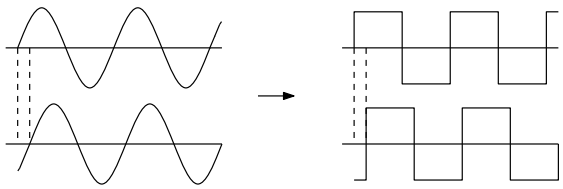


Figure 1.1: Discarding amplitude information

The delay information is then extracted for each pair of microphones from these binary signals and fed to the processing scheme. We implement 2 schemes - a hard-assignment scheme based directly on the delay values and a soft-assignment scheme that is based on the beamforming technique described previously. We show how this approach significantly reduces computational complexity at the expense of a little loss in accuracy. Simulation results are presented for a square array of 9 sensors and agreement between simulation and implementation results are shown for a square array of 4 sensors.

1.5 Structure of the thesis

The rest of the thesis is organized as follows:

- **Chapter 2: Hardware Details**
  This chapter deals with the details of the hardware being used for prototyping the ideas proposed in the thesis. Modifications to traditional microphone array front-end architectures are discussed. The TDC scheme used for finding the delay is also described in this chapter.

- **Chapter 3: TDOA and Chapter 4: DSBF**
  These chapters deal with introducing the TDOA and DSBF algorithms. Details specific to our prototype implementation as well as suggested modifications are listed. The Verilog-based hardware architecture for implementing both the algorithms on the FPGA is described.

- **Chapter 5: Results and Discussion**
  Simulation and implementation results for both the algorithms are presented. The DSBF algorithm performance is compared against existing work in the area.

- **Chapter 6: Conclusions and Scope for Future Work**
  The contributions of this work are described in this chapter. Further directions of work are suggested.

CHAPTER 2

Hardware Details

2.1 Description of the front-end

For the purpose of this project, we use transmitters and receivers that are tuned to 25kHz. The FPGA being used is the Xilinx Zynq 7Z020 that comes as part of the Zedboard kit. This FPGA is clocked at 100MHz. The circuit operates from a supply voltage of 3.3V. We consider a square microphone array using 4 sensors with the side-length of the square being 25mm.

Since the premise of our idea is based on deriving the delay information from the sign bit, the only hardware preprocessing that needs to be done is to hard-limit the received signal appropriately. Consider the architecture for the receiver front-end shown in Figure 2.1. While this figure only shows the components for a single receiver, the same circuit is used for each of the 4 receivers so that a square wave is obtained at the output of each of these.

We note here that the front-end is simpler compared to traditional front-ends used in microphone arrays. A typical front-end consists of a receiver followed by a Variable Gain Amplifier (VGA) to boost the amplitude so that the final amplitude covers the dynamic range of the Analog to Digital Converter (ADC) that follows. The output of the ADC is then the quantized amplitude value at that instant. In large arrays with several microphones, transferring all this data onto the processing platform also becomes a challenge. Time multiplexing is often involved in these cases (Zimmermann and Studer, 2010). On the other hand, in the proposed architecture, the output of each copy of the front-end circuit is a single wire containing the binary signal we desire. Modern FPGAs typically have several I/O pins and there is no need of any intermediate interface circuitry since the output of the binary front-end is at the same voltage level as that used by the FPGA.

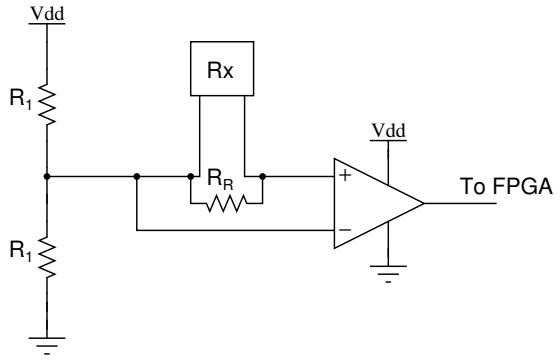


Figure 2.1: Receiver Front-end

Since the receiver is tuned to 25kHz, the voltage that is developed across the terminals of the receiver will be $A\,sin(2\pi 25000t + \phi)$ where A is the amplitude of the received signal and ϕ is its phase. Clearly, the voltage across the receiver varies from -A to A. On the other hand, the FPGA operates from a supply voltage of 3.3V and hence requires its inputs to be between 0 and 3.3V. Hence, we need to level-shift the received signal.

The resistors R_1 help in achieving this DC level shift. The two R_1's form a simple resistor divider, so that the potential at the node in between them becomes $\frac{V_{dd}}{2}$. Since one end of the receiver is held constant at this potential, the positive input to the comparator becomes $\frac{V_{dd}}{2} + A\,sin(2\pi 25000t + \phi)$. This, along with the fact that the other input of the comparator is fixed at $\frac{V_{dd}}{2}$ gives that the output of the comparator will be $\frac{V_{dd}}{2} + \frac{V_{dd}}{2} \times sign(sin(2\pi 25000t + \phi))$, so that it varies from 0 to V_{dd}. Setting V_{dd} to be 3.3V (the supply voltage of the FPGA) then completes the interfacing circuitry. Figure 2.2 shows such a circuit wired on a breadboard.

The problem of estimating the time difference between the two received signals thus transforms into the problem of finding the time difference between the edges of two signals as shown in Figure 2.3.

2.2 Time delay estimation using FPGA

The problem of time delay estimation between edges of the received signals is solved by using a TDC. Since this is merely a prototype to show that our idea works in principle,

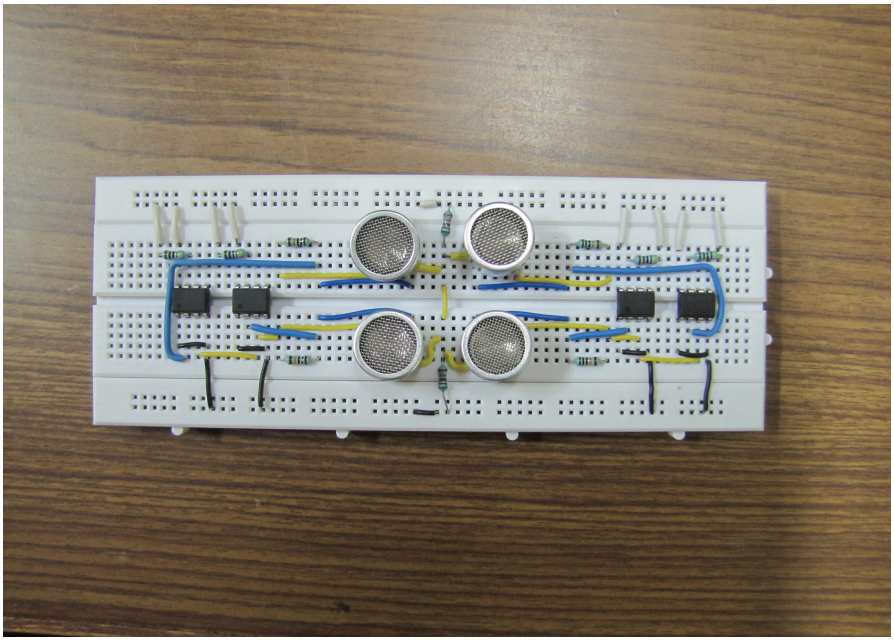


Figure 2.2: Photograph of the circuit on a breadboard

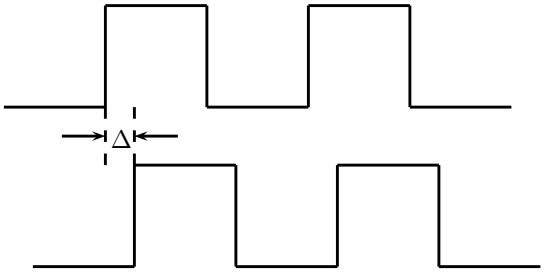


Figure 2.3: Time delay estimation

we employ a simple Nutt interpolation scheme (Balla *et al.*, 2012). Consider the signals shown in Figure 2.3. If we encounter a rising edge of the first signal first, the value of a counter is incremented every clock cycle until a rising edge is seen on the second signal. Conversely, if a rising edge on the second signal arrives first, the counter value is decremented until a rising edge arrives on the first signal. The counter value is reset to 0 once the delay value has been recorded.

For reasons motivated in Section C.2, we need to run the TDC at as high a clock frequency as possible. In other words, the time delay must be quantized such that the resolution of the TDC is minimized. The FPGA runs on a global clock of 100MHz. However, the module associated with the delay estimation can be run at a higher frequency. Implementation results show that a frequency of 400MHz can be realized on hardware. We further employ a phase-shift technique to increase the resolution by a

factor of 4. In this technique, 4 signals are generated at 400MHz (using the DCM IP Core), with relative phase shifts of $0°$, $90°$, $180°$ and $270°$.

With each of these 4 clocks, we run a TDC module using the *same* input signal. Thus, we obtain four delta values for the same signal generated by each of the 4 counters run using the 4 clocks generated as above. We note that the value generated by these 4 counters can always be expressed as belonging to the set (k, $k + 1$). To illustrate this point, consider the situation shown in Figure 2.4 again. In this case, we can see that the counter driven by the first clock will count a value that will be less than that generated by the other 3 counters by 1. Thus, if the count value of the TDC driven by the first clock is k, that of the TDCs driven by the other 3 values can be written as $k + 1$. This observation helps us reduce the hardware requirements. Thus, the TDC enables a time resolution of 625ps corresponding to a frequency of 1600MHz.

The Verilog implementation of this scheme is described in the following section.

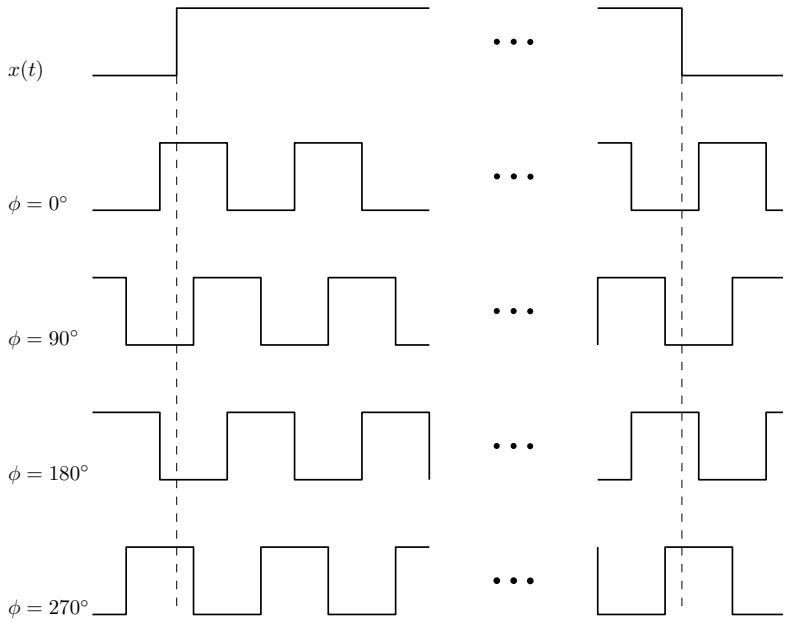


Figure 2.4: 4-phase shifted clock signal scheme to increase TDC resolution

2.3 Verilog Implementation

Since estimating the time delay forms just the first step of the localization scheme, we create a module that implements the 4-phase Nutt interpolation method described

previously in Verilog. Assuming an array of N_{mic} microphones, we need to estimate at least $N_{mic} - 1$ such delay values.

We note that it is inefficient to run separate counters for each instantiation of the module since they essentially become copies of each other. A better way of implementing this could be having 4 common global counters that each instance accesses for their latest values. The corresponding output of each module can then be expressed as a difference assuming that the global counter's overflow is taken care of. However, this has not been implemented in this work. A small simplification has been performed which does allow for minimizing the hardware required for adding the 4 delay values generated from each of the 4 phase-shifted clocks. The theory behind this is explained in Appendix B. Instead of using 4 16-bit counters, it allows us to obtain the result using 1 16-bit counter and 3 2-bit counters.

2.3.1 delayTDC

This module is instantiated for every delay value needed. For the 4 sensor example being considered, there are 3 copies of this module that are created. It operates at 100MHz although it receives the 4 phase-shifted 400MHz clocks from the DCM as well. It returns a 20-bit delay count value such that the LSB offers a time resolution of 625 ps. This module itself has several submodules as can be seen in Figure 2.5.

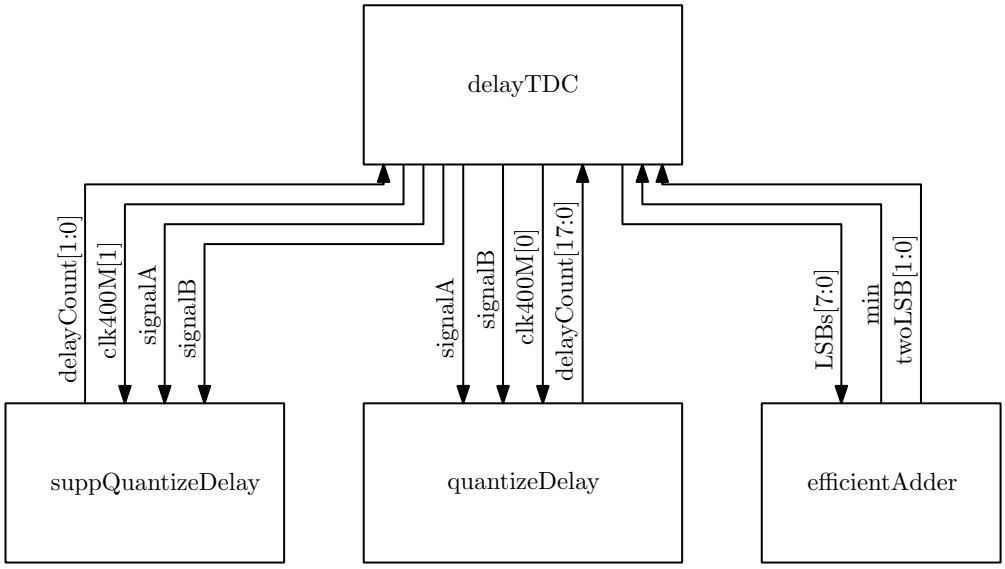


Figure 2.5: Delay TDC Architecture

quantizeDelay

This module operates at 400MHz and counts the number of clock cycles between the edges of the two signals. Although a 16-bit counter should theoretically suffice for our case, we implement an 18-bit counter to account for possible design changes later. This module returns the complete 18 bit value.

suppQuantizeDelay

This module is based on the quantizeDelay module, but it operates with one of the phase shifted versions of the 400MHz clock. However, instead of using a 18-bit counter like the quantizeDelay module, it counts only a 2-bit value and returns the same. Essentially, this module tracks only the 2 LSB bits of the count value. 3 of these modules are instantiated per delayTDC module - 1 for each of the phase-shifted clocks.

efficientAdder

This module receives the 2 LSB values of each of the 4 counters and then returns 2 values - a min value and the last 2 LSBs of the sum. The min value is used to indicate whether the output of the quantizeDelay module was of the form k or $k+1$. Based on the min value that is given here, the delayTDC module appends the appropriate 2 LSB bits to the output of the quantizeDelay module. This module is purely combinatorial.

CHAPTER 3

TDOA

The Time Difference Of Arrival (TDOA) method relies on reliably estimating the difference in arrival times between different sensors. Consider the sensor setup shown in Figure 3.1.

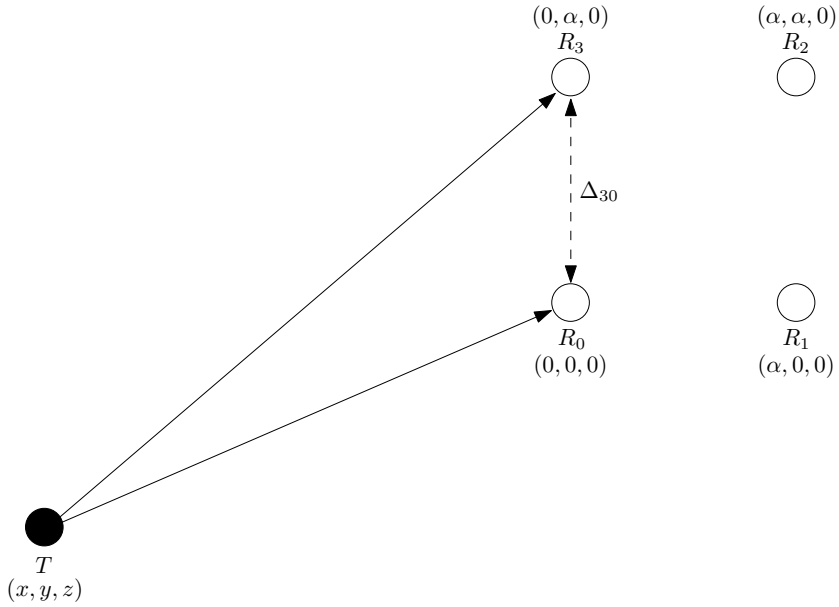


Figure 3.1: Sensor array

We thus have the following equations:

$$
\begin{aligned}
x^2 + y^2 + z^2 &= c^2\tau^2 \\
(x-\alpha)^2 + y^2 + z^2 &= c^2(\tau + \Delta_{10})^2 \\
x^2 + (y-\alpha)^2 + z^2 &= c^2(\tau + \Delta_{30})^2 \\
(x-\alpha)^2 + (y-\alpha)^2 + z^2 &= c^2(\tau + \Delta_{20})^2
\end{aligned}
$$

where c is the speed of sound in air, α is the inter-sensor distance along a side of the square and each Δ_{ij} is the time difference between the arrival of the received signals at sensors i and j.

Solving these equations gives us the following expressions for τ, x and y. The complete derivation is presented in Appendix A.

$$\tau = \frac{\Delta_{20}^2 - \Delta_{10}^2 - \Delta_{30}^2}{2(\Delta_{10} + \Delta_{30} - \Delta_{20})} \tag{3.1}$$

$$x = \frac{\alpha}{2} - \frac{c^2}{2\alpha} \frac{\Delta_{10}(\Delta_{20} - \Delta_{30})(\Delta_{20} + \Delta_{30} - \Delta_{10})}{\Delta_{10} + \Delta_{30} - \Delta_{20}} \tag{3.2}$$

$$y = \frac{\alpha}{2} - \frac{c^2}{2\alpha} \frac{\Delta_{30}(\Delta_{20} - \Delta_{10})(\Delta_{20} + \Delta_{10} - \Delta_{30})}{\Delta_{10} + \Delta_{30} - \Delta_{20}} \tag{3.3}$$

3.1 Implementation Details

3.1.1 Spatial aliasing

Since the signal that we expect to receive is a continuous 25kHz wave, given a frame, we can only estimate relative delays in the range $[-\pi, \pi]$. However, if this is to be valid for all possible transmitter locations, we would need the sensors to have a maximum inter-sensor distance of $\frac{c}{2f_{signal}} = 6.8$mm. This cannot be achieved because the diameter of the sensors is 16.2mm. The sensors are kept at a distance of 25mm from each other because this leads to some computational simplifications. The downside of doing so is a reduced scan-area which the transmitter must be restricted to lie in. Moving the transmitter outside this area results in wrapping around of the output (a clear artifact of aliasing).

3.1.2 Reduction in sampling rate

Since we hard-limit the received signal, the influence of noise is indirect. Near the zero-crossings of the signal, noise tends to push the signal beyond the threshold sooner or later randomly. This manifests itself as jitter in the edge. This is analyzed in greater detail in Appendix C.1.

Upon testing with actual sensors, it was validated that the noise did, in fact, create significant jitter at the signal edges. Therefore, a 21-tap Kaiser-Bessel low-pass filter was implemented using the Xilinx IP Core to smooth out the delta values. Of the 20-

bit value that the delayTDC module was supposed to return (thereby realizing a time resolution of 625ps), only the 13 MSB bits stayed constant. This implies an error 128 times larger than the expected error. In other words, the highest possible sampling rate obtained was 12.5MHz, 128 times lesser than the expected sampling rate of 1600MHz. As a result, the calculated coordinates for the source were found to be highly inaccurate. Reexamining the equations for τ, x and y (3.1, 3.2, 3.3) explains why this is so. The denominator in all 3 cases is $\Delta_{30} + \Delta_{10} - \Delta_{20}$ which can be rewritten as $\Delta_{10} - \Delta_{23}$, which is the difference of two values that are numerically very close to each other. Thus, unless this value can be estimated accurately, the values derived using this expression will continue to be inaccurate.

3.1.3 Conversion to spherical coordinates

Rewriting the equations in spherical coordinates, we have

$$\begin{aligned} r &= cT \\ -2\alpha r cos\theta cos\phi + \alpha^2 &= 2c^2\tau\Delta_{10} + c^2\Delta_{10}^2 \\ -2\alpha r sin\theta cos\phi + \alpha^2 &= 2c^2\tau\Delta_{30} + c^2\Delta_{30}^2 \\ -2\alpha r cos\theta cos\phi - 2\alpha r sin\theta cos\phi + 2\alpha^2 &= 2c^2\tau\Delta_{20} + c^2\Delta_{20}^2 \end{aligned}$$

Since we have established that with the given sampling rate of 12.5MHz, it will be hard to estimate τ accurately, we assume a constant value of τ so that r becomes equal to 1m. What this implies is that the values generated using this assumption will be projected onto a sphere of radius 1. In other words, by fixing the radius to a particular value, we aim to estimate the direction accurately. Thus, since $x = rcos\theta cos\phi$ and $y = rsin\theta cos\phi$, we get accurate values for x and y given by the following equations:

$$x = \frac{\alpha}{2} - \frac{2c\Delta_{10} + c^2\Delta_{10}^2}{2\alpha} \tag{3.4}$$

$$y = \frac{\alpha}{2} - \frac{2c\Delta_{30} + c^2\Delta_{30}^2}{2\alpha} \tag{3.5}$$

The values of x and y thus obtained are then displayed on a VGA monitor.

3.2 Verilog implementation

In order to implement this method, the architecture is modularized and designed as shown in Figure 3.2. The functions of each module are discussed in the following sections.

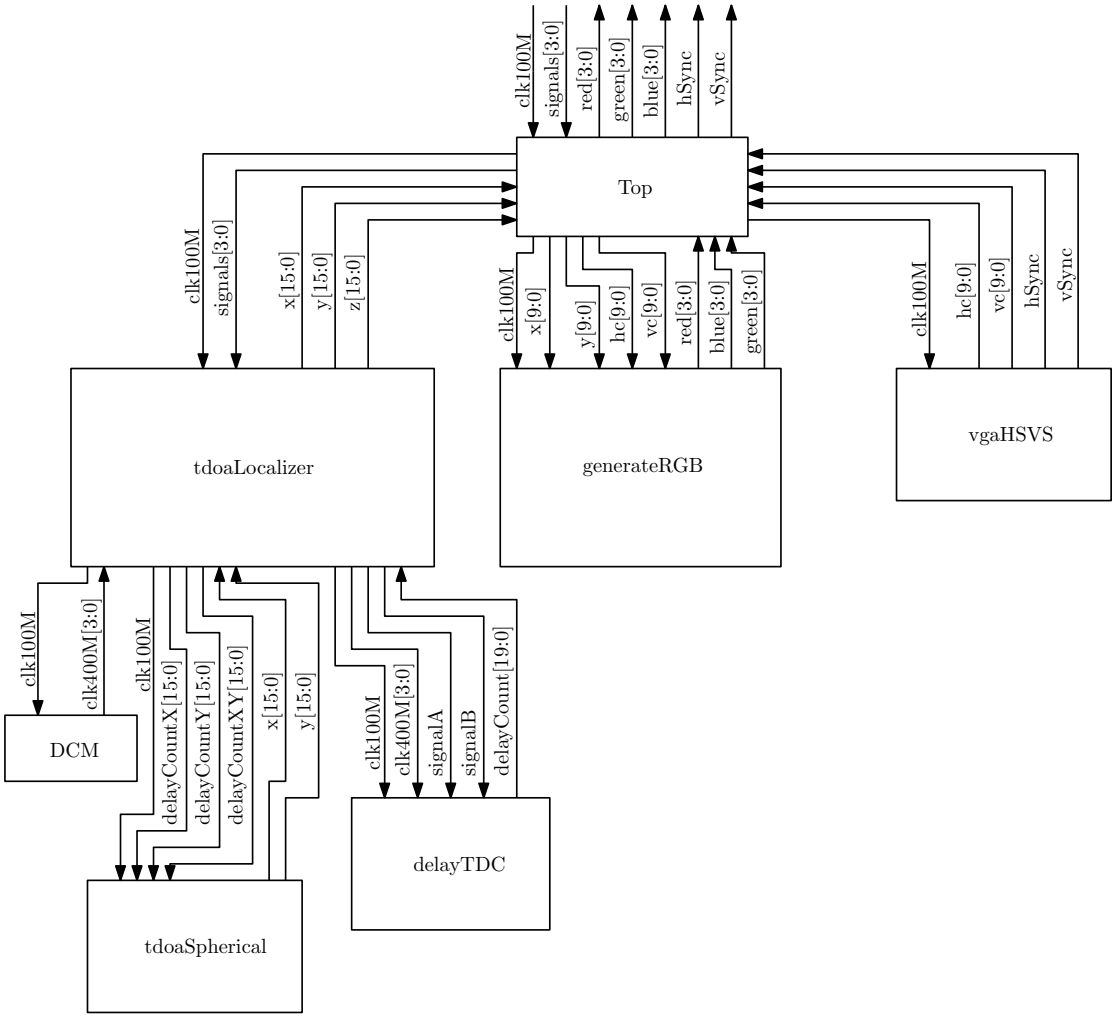


Figure 3.2: Architecture level design

3.2.1 Top

This module acts as an interface with the receiver front-end. It is responsible for ensuring that all the other modules are synchronized correctly. It receives the signals from the front-end and passes them on to the TDOALocalizer module. Further, it passes on the calculated values of x and y from the TDOALocalizer module to the generateRGB module. This module operates using the 100MHz global clock.

3.2.2 TDOALocalizer

This module performs several operations. Firstly, it passes on the 100MHz clock to the DCM and receives the 4 phase-shifted versions of the 400MHz clock from it. All of these clock signals, along with the 2 appropriate signals chosen from the 4 received signals are then passed on to the delayTDC module. For example, the delayTDCX module will receive signals 0 and 1 and the delayTDCY module will receive signals 0 and 3. It receives a 20-bit value from each of the delayTDC modules corresponding to Δ_{10}, Δ_{20} and Δ_{30}.

This module also passes on the received Δ_{ij} values on to the tdoaSpherical module and receives the x and y values in return. It operates at a frequency of 100MHz.

3.2.3 DCM

This module is an instantiation of the Xilinx Digital Clock Manager IP Core. It is responsible for generating the 4 phase-shifted 400MHz clocks required for the delay estimation.

3.2.4 tdoaSpherical

This module receives the different Δ_{ij}'s and in turn, calculates the values of x and y using equations 3.4 and 3.5. It operates at 100MHz.

3.2.5 vgaHSVS

This module is responsible for generating the signals "hSync" and "vSync" that are used for synchronizing the horizontal and vertical beams of the VGA display. It also generates the values "hc" and "vc" which keep track of which pixel information is being displayed at that instant. In our implementation, we have chosen to display 640×480 pixels. This requires a pixel clock of 25MHz. In other words, the "hc" value increments at a rate of 25MHz until it reaches its limit, at which point the "vc" value is incremented

and “hc” is reset to 0. This 25MHz clock is derived from the 100MHz clock provided as an input to the module.

3.2.6 generateRGB

This module is responsible for visualizing the source location on the VGA display. The received values of x and y are used so that when the pixel corresponding to the coordinate (x, y) is about to be displayed, this module sets the value of green to 15. All the other values are kept 0. Thus, the source location is marked by a green dot. In order to make sure that the dot is visible, it is expanded to a square of size 4×4 pixels. This module also ensures that the X and Y axes are drawn in blue. It operates at a frequency of 100MHz.

CHAPTER 4

DSBF

The Delay and Sum BeamForming (DSBF) approach is slightly different from the TDOA approach in that it does not inherently depend on estimating the delays between the received signals. An illustration of the approach can be seen in Figure 4.1.

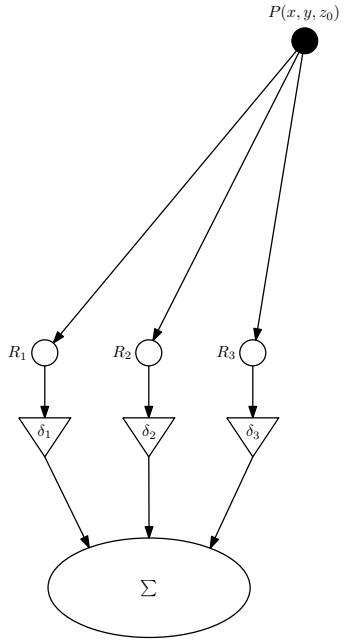


Figure 4.1: DSBF Principle

In this approach, depth information cannot be retrieved. Hence, it is assumed that a plane at a fixed depth is being scanned. This plane is then split into a grid along the X and Y axes. For each pixel in this grid, the expected delay value at each sensor is precalculated. This can be done since the coordinates of both the sensor and the pixel are known. Mathematically, for each sensor R_i and each pixel P in the scan plane, we calculate the path delay $\Delta_{c,i}(P) = \frac{f_s}{c}||P - R_i||$.

The received signal at each sensor is then shifted by this calculated value of $\Delta_{c,i}(P)$. For pixel locations where the source is indeed present, shifting in this manner will make the shifted received signals at all the sensors align in phase so that the energy of the sum of these shifted signals will be maximum. Conversely, for pixel locations with no source

present, this shifting will ensure that the signals don't align, so that the energy of the sum of these shifted signals will not be as high. Mathematically, the sum of the delayed signals can be expressed as

$$s(P)[k] = \sum_{i=1}^{N_{mic}} s_i[k - \Delta_{c,i}(P)]$$

where $s_i[k]$ is the signal received at sensor R_i. This can then be used to calculate the energy of the signal as

$$I(P) = |s(P)[k]|^2$$

Clearly, points with high values of $I(P)$ are more likely to contain the source, and those with low values are less likely to do so. $I(P)$ can thus directly be displayed onto a screen in order to visualize the plane being scanned.

4.1 Implementation Details

Considering the fact that we can only estimate time differences in our system, we alter the definition of the calculated delay time. We express all such calculated delay times with the sensor at the origin as the reference. Thus the definition of $\Delta_{c,i}(P)$ is modified as $\Delta_{c,i}(P) = \frac{f_s}{c}(||P - R_i|| - ||P - R_0||) = \frac{f_s}{c}(||P - R_i|| - ||P||)$ since R_0 is considered to be the origin. By definition, $\Delta_{c,0} = \frac{f_s}{c}(||P - R_0|| - ||P - R_0||) = 0$. We have been using the notation Δ_{ij} to denote the observed relative delay between the signals received at sensors i and j. We note that the observed delay at sensor i is $\Delta_{o,i} = \Delta_{i0}$. Similar definitions are used in (Zimmermann and Studer, 2010) to reduce memory requirements.

In our particular implementation, we know that the received signals $s_i[k]$ are binary, i.e. $s_i[k] \epsilon \{-1, +1\}$. Thus the expression for the sum can be simplified further and expressed in terms of the estimated time lag between the signals. Consider the situation shown in Figure 4.2.

It is evident that a value of $\Delta_{c,i}(P)$ that is equal to $\Delta_{o,i}$ will give maximum overlap and hence maximum energy for the sum of the delayed signals. However, in general,

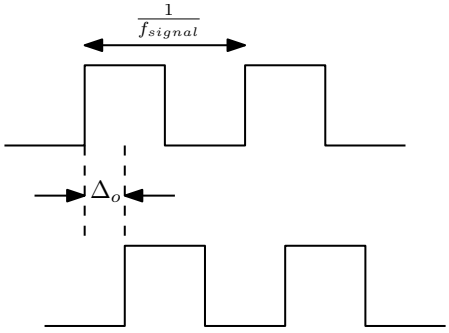


Figure 4.2: Typical received signals

the value of $\Delta_{c,i}(P)$ will not be exactly equal to $\Delta_{o,i}$. Thus, for a general calculated value of $\Delta_{c,i}(P)$, if the second signal is shifted, the resultant delay between the two signals will now be $\delta_i(P) = \Delta_{o,i} - \Delta_{c,i}(P)$. If the signals are now added up with the delay between them being $\delta_i(P)$, we can show that the energy of the total overlap in one period will be

$$I_i(P) = 4(1 - 2f_{signal}|\delta_i(P)|) \tag{4.1}$$

With varying values of $\delta_i(P)$, this value will also vary, peaking for $\delta_i(P) = 0$ implying $\Delta_{c,i}(P) = \Delta_{o,i}$ as expected.

Given an array of sensors, we can calculate all the $\Delta_{c,i}$'s on the FPGA and estimate the $\Delta_{o,i}$'s by reusing the delayTDC module. Thus, the $\delta_i(P)$'s can be computed for different pixel coordinates. Places where the value of $\delta_i(P)$ goes to zero indicates maximal overlap of the signal at sensor i with the signal at the origin. We note that $\delta_i(P) = \delta_j(P)$ indicates maximal overlap between signals at sensors i and j irrespective of the actual value of $\delta_i(P)$. For the absolute maximal overlap, all the $\delta_i(P)$'s must go to 0.

In the specific case of 4 receivers, the signal obtained by summing the delayed version of the 4 received signals can have a maximum of 4 different amplitude levels (refer Figure 4.3). Calculating the energy of such a signal would involve sorting the delay values which is hard to implement on an FPGA and are generally not friendly to scaling schemes.

Thus, instead of evaluating the intensity directly, we calculate the pairwise intensity and then rewrite the total intensity in terms of these pairwise intensities. Recall that the intensity of a pair of square waves can be expressed in terms of their relative delay as

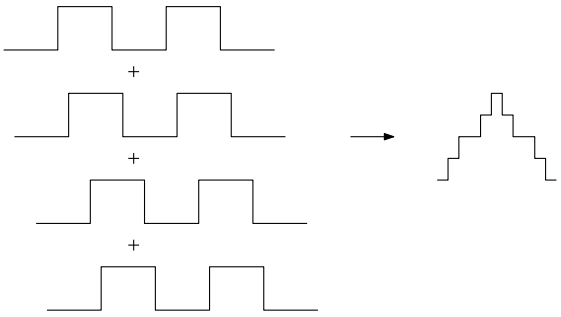


Figure 4.3: Example of summing 4 signals

shown in Equation 4.1. The intensity of the sum of all the waves can be written as

$$I(P) = \frac{1}{T}\int_0^T (x_1(t) + x_2(t) + x_3(t) + x_4(t))^2 dt$$

$$\therefore I(P) = \frac{1}{T}\int_0^T \left(x_1^2(t) + x_2^2(t) + x_3^2(t) + x_4^2(t) + 2x_1(t)x_2(t) + 2x_1(t)x_3(t) + \right.$$
$$\left. 2x_1(t)x_4(t) + 2x_2(t)x_3(t) + 2x_2(t)x_4(t) + 2x_3(t)x_4(t)\right) dt$$

$$\therefore I(P) = \frac{1}{T}\left(\sum_{i=1}^{3}\sum_{j=i+1}^{4}\int_0^T (x_i(t) + x_j(t))^2 dt - 2\sum_{i=1}^{4}\int_0^T x_i^2(t) dt\right)$$

In general, with N_{mic} microphones

$$I(P) = \frac{1}{T}\left(\sum_{i=1}^{N_{mic}-1}\sum_{j=i+1}^{N_{mic}}\int_0^T (x_i(t) + x_j(t))^2 dt - N_{mic}(N_{mic}-2)\sum_{i=1}^{N_{mic}}\int_0^T x_i^2 dt\right)$$

We note that $\int_0^T x_i^2 dt$ remains constant in our case since the signal will be a constant 25kHz wave.

4.2 Verilog Implementation

The implementation of this algorithm requires us to know the observed and calculated delay values for each receiver, $\Delta_{o,i}$ and $\Delta_{c,i}$ respectively. The observed delta values

are obtained using the same module as in the TDOA method, so that once again, an effective sampling rate of 12.5MHz is achieved. Choosing the scan plane to be at a depth of $z = 2$m from the receiver array, we have a scan area of (-0.25m, 0.25m) in both the X and the Y directions. The block diagram for the architecture is shown in Figure 4.4. The functions of modules not discussed previously are discussed here:

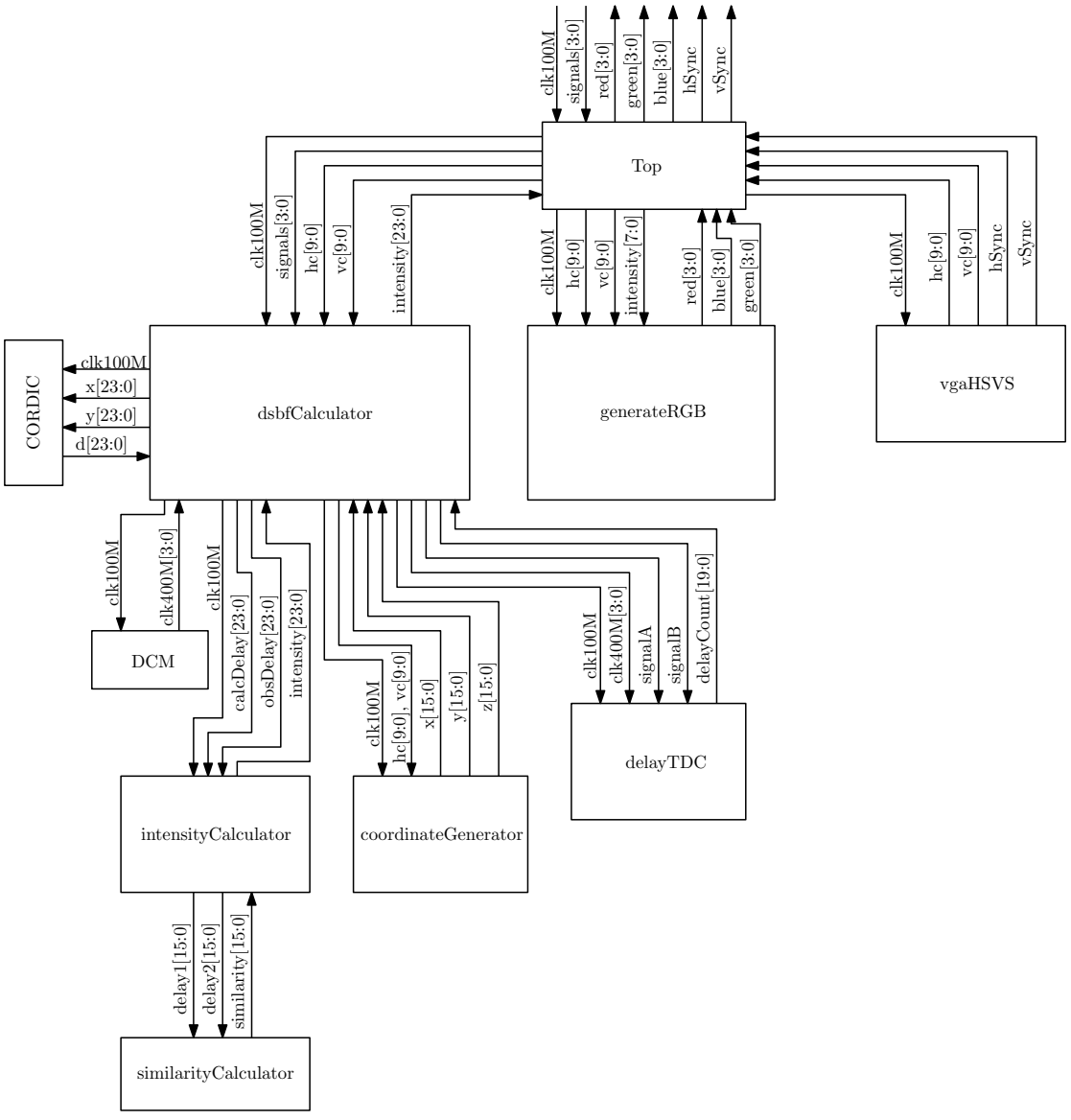


Figure 4.4: Architecture for implementing the DSBF algorithm

4.2.1 dsbfCalculator

This module is similar to the module 3.2.2, but it differs from the TDOALocalizer module in that it receives the coordinate values from the coordinateGenerator module and

passes these on to the CORDICs. The corresponding distance value is then passed on to the intensityCalculator module from which it receives the intensity value in exchange. This module operates at 100MHz.

4.2.2 coordinateGenerator

This module receives inputs from the vgaHSVS module through the dsbfCalculator module and generates the coordinates of the pixel being scanned with respect to each of the 4 receivers. Coordinates are generated in a 16-bit format with 5-bit integer and 11-bit fractional part. The latency of calculation is accounted for in such a way that after all the calculations are performed, the intensity value generated corresponds exactly to the pixel being displayed at that moment (just-in-time approach). This is done to reduce memory utilization.

This module operates at 100MHz but the coordinate values change only at 25MHz since 4 cycles are needed to output the coordinates of the pixel wrt each of the 4 sensors.

4.2.3 CORDIC

These modules are instantiations of the Xilinx CORDIC IP Core used for calculating the distance from the pixel being scanned to each of the receivers. Each instance, given inputs x and y, calculates $\sqrt{x^2 + y^2}$. 2 instances of these are chained together to thereby evaluate $\sqrt{x^2 + y^2 + z^2}$.

This module runs at a clock speed of 100MHz and has a latency of 32 cycles with a throughput of 1 value per cycle. The final result has 5 bits for the integer part and 19 bits for the decimal part.

4.2.4 intensityCalculator

This module aggregates the distance values for a pixel with respect to each of the 4 sensors, calculates the corresponding delay value by accounting for the $\frac{f_s}{c}$ scaling factor and then passes these values on to instances of the similarityCalculator module. The

final intensity value is then computed and squared in order to increase the dynamic range.

This module runs at 100MHz and has a latency of 7 cycles.

4.2.5 similarityCalculator

This combinatorial module performs the operations required for finding the intensity of the sum of 2 square waves given the delay between them.

4.2.6 generateRGB

This module is modified so as to generate an RGB intensity map based on the 6-bit value of the intensity that it receives. The colour gradually changes from red to green to blue as the intensity decreases from its max value to its min value. Since the intensity is calculated as 24-bit number, it is possible to control the 6-bits used for displaying the output so that the dynamic range of the output is maximized.

CHAPTER 5

Results and Discussion

We discuss results obtained using both of the algorithms with the proposed approach in this chapter. Simulation results are compared against those obtained from implementation on a Xilinx Zynq 7Z020 FPGA board. Verilog simulations are performed using the free tool IVerilog.

5.1 TDOA

5.1.1 Simulation Results

We consider a single source to be present at (-0.15, 0.25, 2). In spherical coordinates, this can be written as (2.02, -59.03°, 8.29°). We then generate the signals that the FPGA should ideally receive by manually entering the delay values in the simulation with a very high degree of accuracy. The waveforms marked "signals" in Figure 5.1 show the signals after they have been hard-limited. The waveforms t_{10}, t_{11} and t_{01} indicate the value of Δ_{10}, Δ_{20} and Δ_{30} respectively estimated using a TDC operating at 12.5MHz.

The module reports values of x and y that are scaled up by a factor of 512. Hence, the actual value of x and y is (-0.068, 0.119). In fact, since we assume a value of 1 for r in our algorithm, and seek to estimate θ and ϕ accurately, comparing the estimated values of these parameters is a good measure of the algorithm's validity. Using $r = 1$ and (x, y) = (-0,068, 0.119), we get $\theta' = tan^{-1}\left(\frac{y}{x}\right)$ = -60.15° and $\phi' = sin^{-1}\left(\frac{\sqrt{x^2+y^2}}{r}\right)$ = 7.89° which are very close to the expected values. An analysis of the impact of quantization noise on the performance of the algorithm can be seen in C.2.

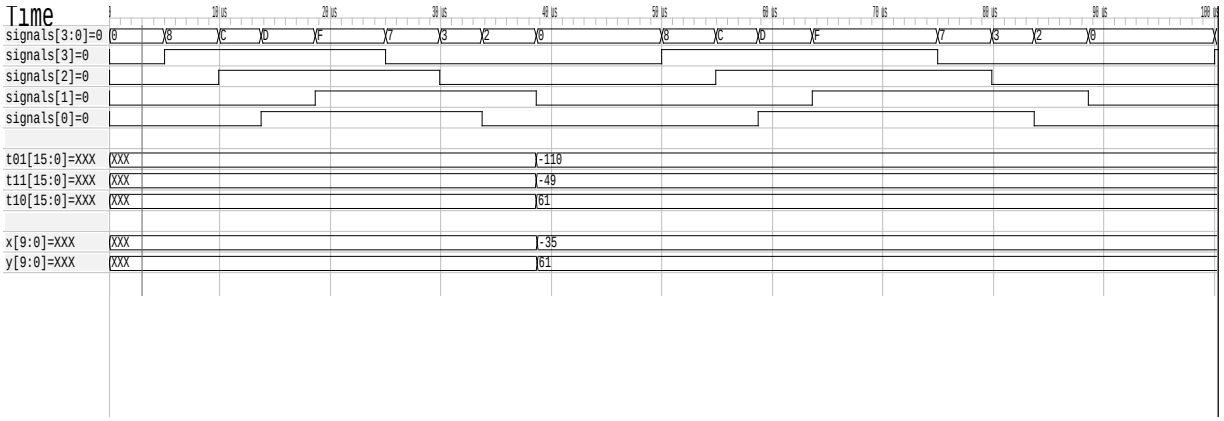


Figure 5.1: Simulation results for the spherical TDOA approach

Clock frequency	100MHz
Slices	465 (3%)
Slice flip-flops	1843 (1%)
LUTs occupied	1036 (1%)
DSP Units	23 (10%)

Table 5.1: Resource utilization for the TDOA algorithm

5.1.2 Implementation Results

The result is displayed on a VGA display with a resolution of 640×480 with a refresh rate of 60Hz. This algorithm generates an updated value of the source location at a rate equal to f_{signal}, which is 25kHz in our case. Since this is far greater than the refresh rate of the display, there are no issues in achieving real-time performance. Table 5.1 gives the resource utilization for the algorithm.

As the distance of the source from the array increases, since the SNR decreases, the source location estimates start becoming increasingly noisy. The stationary source location estimate observed when the source is close to the array starts moving randomly about the actual location of the source.

Figure 5.2 shows a typical output as seen on a VGA display. The green dot marks the location of the source.

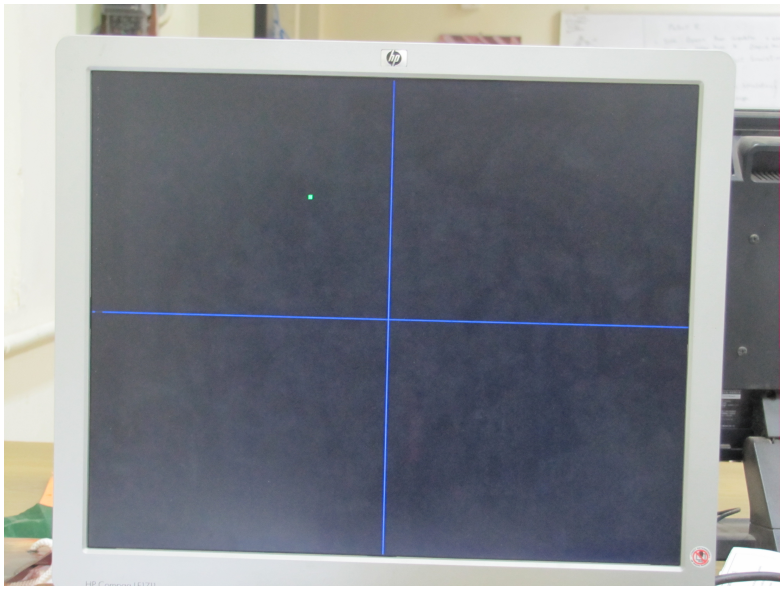


Figure 5.2: TDOA output as seen on a VGA display

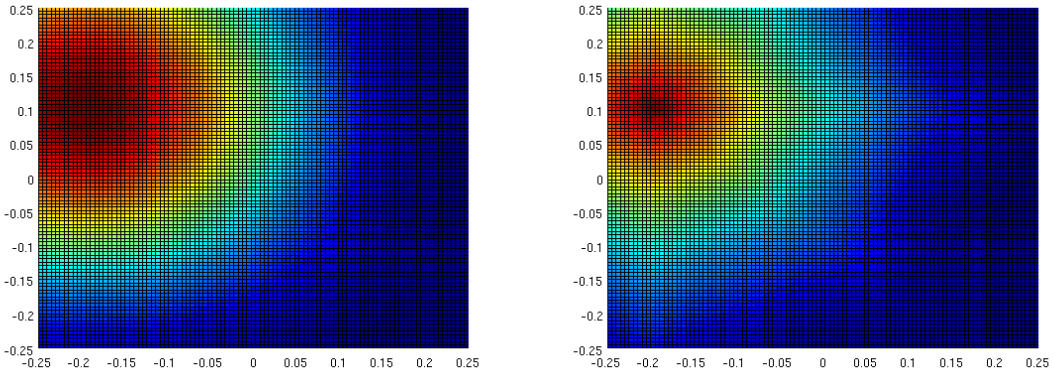
5.2 DSBF

In the TDOA method, the estimated time-delay values were used directly in the computation of the source location. Thus, hard-limiting made sense intuitively, because it was the phase difference between the received signals rather than the amplitude difference that captured the delay information.

Claiming such a thing for the DSBF method, on the other hand, is incorrect, since it appears to use the amplitude information as well. Hence, it is important to check the validity of the method once we hard-limit the received signal. Although the received amplitude will be approximately the same at each receiver if there is only one source, this is not true in the case of multiple transmitters. Indeed, even with 2 transmitters, the amplitude is modulated by a factor of $cos\left(\omega\left(\frac{d_2 - d_1}{2c}\right)\right)$ where d_1 and d_2 are the distances of the two transmitters from the receiver under consideration. Simulation results show that this term varies significantly even if the 2 sensors are placed close to each other, so much so that it might even introduce an inversion in the received signal.

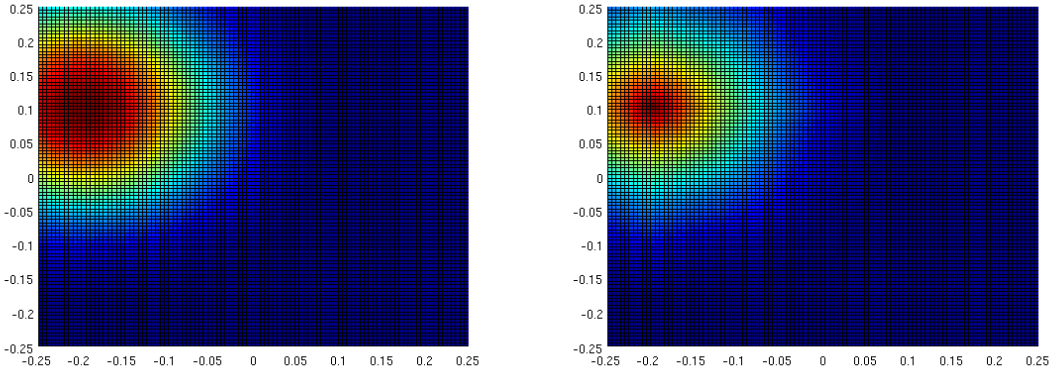
5.2.1 Simulation Results

Figures 5.3 and 5.4 show simulation results with 4 and 9 receivers respectively and a single transmitter at (-0.2, 0.1, 2).



a b

Figure 5.3: Simulation results for DSBF using 4 receivers and 1 transmitter



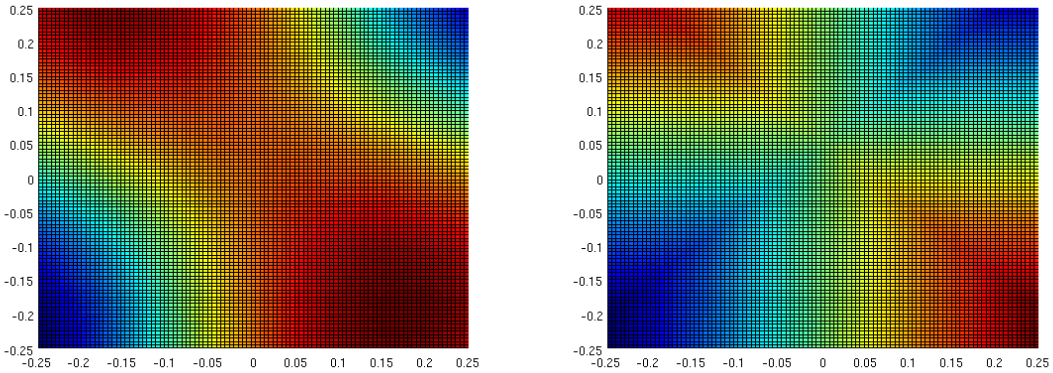
a b

Figure 5.4: Simulation results for DSBF using 9 receivers and 1 transmitter

It is clear from Figures 5.3b and 5.4b and their similarity with Figures 5.3a and 5.4a respectively that hard-limiting does not take away all the information about the location of the source. While the intensity plots lose their circular gradient about the source location, the actual location of the source remains the same. Since this is the information of primary interest, we conclude that hard-limiting can be used with the DSBF technique as well.

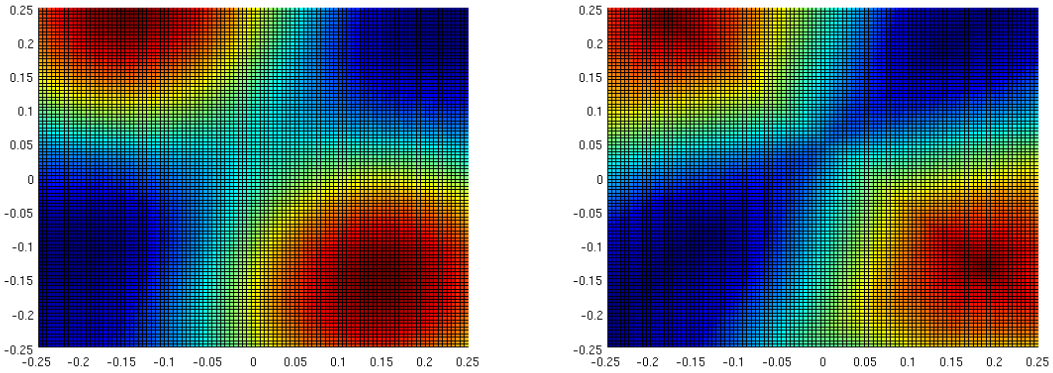
Figure 5.5 shows the simulation results with 4 receivers whereas Figure 5.6 shows the results for the source at the same location, but using an array of 9 receivers. In both cases, 2 transmitters are present at (-0.15, 0.25, 2) and (0.15, -0.15, 2).

Figures 5.5b and 5.6b and their resemblance to Figures 5.5a and 5.6a respectively indicate that the algorithm can be used in case of multiple sources as well. We must



a b

Figure 5.5: Simulation results for DSBF using 4 receivers and 2 transmitters



a b

Figure 5.6: Simulation results for DSBF using 9 receivers and 2 transmitters

note here, however, that the simulation output for 4 receivers gives the same output as shown in Figure 5.5b for a large number of source locations. The reason behind this behaviour is unclear.

5.2.2 Implementation Results

The system was designed for a refresh rate of 60Hz, thereby meeting the real-time requirement. Table 5.2 gives the resource utilization for the algorithm. Figure 5.7 shows a typical output as seen on a VGA display. The blue regions are where the intensity is minimum. The transition of blue to green to red is in the order of increasing intensity values at those coordinates.

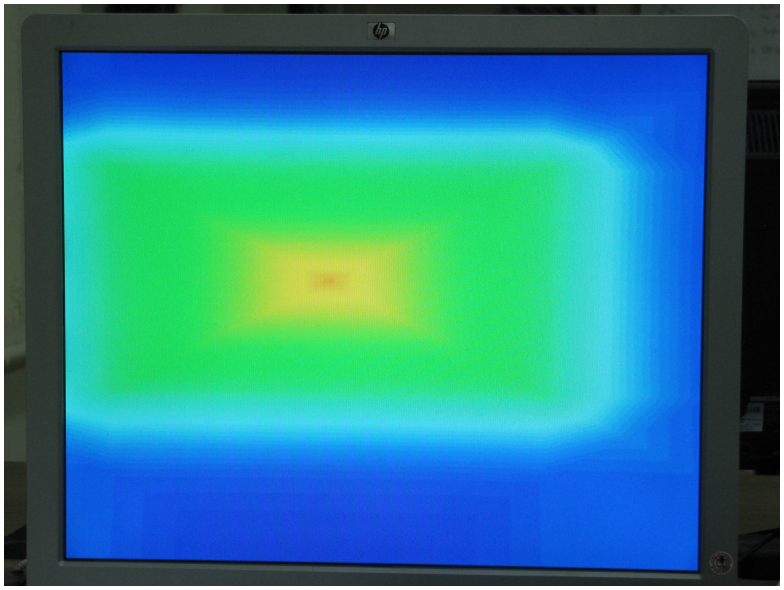


Figure 5.7: DSBF result as seen on a VGA display

Clock frequency	100MHz
Slices	2124 (15%)
Slice flip-flops	7252 (6%)
LUTs occupied	6746 (12%)
DSP Units	34 (15%)

Table 5.2: Resource utilization for the DSBF algorithm

The only other comparable work in this category (Zimmermann and Studer, 2010) implements the DSBF algorithm on an FPGA directly. Consequently, their implementation is of the order $\mathcal{O}(XYN_{mic}L)$ where

X is the number of pixels along the X-axis,

Y is the number of pixels along the Y-axis,

N_{mic} is the number of microphones in the array,

L is the length of the frame (in samples) used for finding the intensity

Although our algorithm does not reproduce the exact output of the DSBF algorithm, it does reduce the computational cost. Since we evaluate the intensity by considering the received signals pairwise, the order is $\mathcal{O}(XYN_{mic}^2)$. Thus, there is a net improvement of the order of $\frac{L}{N_{mic}}$. Typically, L is an order of magnitude higher than N_{mic}. In (Zimmermann and Studer, 2010), for example, $L = 256$ and $N_{mic} = 32$ implying an $8\times$ improvement in performance.

CHAPTER 6

Conclusions and Scope for Future Work

6.1 Contribution of the thesis

We have presented a novel approach towards the problem of source localization. The proposed technique was simulated as well as implemented in hardware using Verilog and good agreement was observed.

The major contributions of the work can be listed as follows:

1. **Simplification of front-end circuitry**

   Since our analysis depends only on the sign of the received signal and not the actual amplitude value at any given instant, the variable gain amplifier is replaced by a comparator thereby reducing the circuit complexity. Further, our system does not require the use of ADCs as part of the front-end as the sign bit is only a binary signal. Multiplexing in order to transmit data is avoided since the data for each receiver is transmitted over just 1 wire.

2. **Simplification of DSBF computation**

   Considering only square waves in our analysis allowed us to express the intensity of the sum of a pair of such waves in terms of their relative delay and time-period. Further, the intensity of the sum of multiple such signals can be expressed in terms of such pairwise intensities, each of which can be computed easily.

3. **Validation of proposed approach on hardware**

   The proposed modifications to both the front-end and the algorithm were implemented on a Xilinx Zynq 7Z020 FPGA. Better than real-time performance were achieved in both the TDOA and DSBF algorithms. The output was displayed on a VGA monitor with a resolution of 640×480 at a refresh rate of 60Hz.

6.2 Scope for Future Work

1. Noise is a concern especially because of the non-trivial nature of the analysis involved. Preliminary investigations into the impact of random noise on the performance of the system are presented in Appendix C. However, a more thorough analysis will provide better insights into effectively tackling noise. Modifying the comparator so that it acts like a Schmitt trigger could possibly help reduce the impact of low-amplitude noise near the zero-crossings of the signal thereby improving performance.

2. We have only handled multiple point sources as test cases. Increasing the size of the array can help in identifying the shape of the source as well (Zimmermann and Studer, 2010). It is yet to be verified whether the shape of the source will be maintained once amplitude information is lost.

3. The proposed techniques should ideally work even if the sound source is non-sinusoidal. Reducing the frequency of the source may also help in removing the effects of spatial aliasing at some expense to spatial resolution.

APPENDIX A

Derivation of source coordinates using delay values

We are required to find x, y and z given the equations

$$x^2 + y^2 + z^2 = c^2\tau^2 \tag{A.1}$$

$$(x-\alpha)^2 + y^2 + z^2 = c^2(\tau + \Delta_{10})^2 \tag{A.2}$$

$$x^2 + (y-\alpha)^2 + z^2 = c^2(\tau + \Delta_{30})^2 \tag{A.3}$$

$$(x-\alpha)^2 + (y-\alpha)^2 + z^2 = c^2(\tau + \Delta_{20})^2 \tag{A.4}$$

where c is the speed of sound in air and Δ_{ij} is the time difference between the signals received at sensors i and j.

Subtracting A.1 and A.2 from A.3 and A.4 respectively, we get

$$\alpha^2 - 2y\alpha = c^2(2\tau\Delta_{30} + \Delta_{30}^2) = c^2(\Delta_{20} - \Delta_{10})(2\tau + \Delta_{10} + \Delta_{20}) \tag{A.5}$$

Solving for τ, we get

$$2\tau(\Delta_{10} + \Delta_{30} - \Delta_{20}) = \Delta_{20}^2 - \Delta_{10}^2 - \Delta_{30}^2$$

$$\therefore \tau = \frac{\Delta_{20}^2 - \Delta_{10}^2 - \Delta_{30}^2}{2\,(\Delta_{10} + \Delta_{30} - \Delta_{20})} \tag{A.6}$$

Resubstituting this value of τ in A.5, we have

$$\alpha^2 - 2y\alpha = c^2\frac{\Delta_{30}(\Delta_{20} - \Delta_{10})(\Delta_{20} + \Delta_{10}) + \Delta_{30}^2(\Delta_{10} - \Delta_{20})}{(\Delta_{10} + \Delta_{30} - \Delta_{20})}$$

$$\therefore \alpha^2 - 2y\alpha = c^2\frac{\Delta_{30}(\Delta_{20} - \Delta_{10})(\Delta_{10} + \Delta_{20} - \Delta_{30})}{(\Delta_{10} + \Delta_{30} - \Delta_{20})}$$

$$\therefore y = \frac{\alpha}{2} - \frac{c^2}{2\alpha}\frac{\Delta_{30}(\Delta_{20} - \Delta_{10})(\Delta_{10} + \Delta_{20} - \Delta_{30})}{(\Delta_{10} + \Delta_{30} - \Delta_{20})} \tag{A.7}$$

The symmetry of the system with respect to x and y can now be used to directly write the solution for x. The steps for the derivation remain the same.

$$\therefore x = \frac{\alpha}{2} - \frac{c^2}{2\alpha}\frac{\Delta_{10}(\Delta_{20} - \Delta_{30})(\Delta_{30} + \Delta_{20} - \Delta_{10})}{(\Delta_{10} + \Delta_{30} - \Delta_{20})} \tag{A.8}$$

Since the inter-sensor distance α is kept small, and since we expect the source coordinates to be much greater than α, it is clear that the $\frac{\alpha}{2}$ term does not contribute much to the actual answer and can be neglected for approximation.

We note here that the denominator term in all of the expressions A.6, A.8 and A.7 are differences of very similar physical quantities and hence need to be found with high accuracy. Thus, correctly determining x, y and τ depends on correctly estimating the denominator term. On conversion to spherical coordinates, however, we notice that

$$tan(\theta) \approx \frac{\Delta_{30}(\Delta_{20} - \Delta_{10})(\Delta_{10} + \Delta_{20} - \Delta_{30})}{\Delta_{10}(\Delta_{20} - \Delta_{30})(\Delta_{20} + \Delta_{30} - \Delta_{10})} \tag{A.9}$$

can be calculated accurately. Thus, if we concede on estimating τ accurately, θ and ϕ ,i.e., the direction, can be determined accurately.

APPENDIX B

Efficiently computing the sum of 4 numbers

We know that the four numbers to be added all belong to the set $\{k, k+1\}$. Further, due to the cyclic nature of the system, we are guaranteed that the minimum number of toggles in the LSBs of the 4 numbers will be at most 1. For example, let the 4 numbers be denoted by a, b, c and d. Further, let the subscript i to any of these numbers denote the corresponding LSB of that number so that a_0 indicates the LSB of a and so on.

Due to the cyclic nature of the system, we are guaranteed that $a_0b_0c_0d_0$ can never be of the form 1010. It can, however, take the form 1011 since, starting from 0 and moving cyclically, there is only 1 toggle. In fact, the only disallowed combinations for $a_0b_0c_0d_0$ are 0101 and 1010.

Now, we also know that

$$a + b + c + d = 4k + m$$

where m is the number of values of the form $k + 1$. We can determine which of a, b, c and d are of the form $k + 1$ merely by looking at a_1a_0, b_1b_0, c_1c_0 and d_1d_0. Hence, the sum can be found if only one of a, b, c and d is known exactly and the 2 LSBs of the other 3 are known.

As an example to illustrate this claim, let $a = d = 22$ and $b = c = 21$. In 8-bit binary, we have $a = d = (00010110)_2$ and $b = c = (00010101)_2$. Since the claim is that the sum $4k + m$ can be obtained merely by having only 1 value known exactly, without loss of generality, we can assume that we know a fully, and know b_1b_0, c_1c_0 and d_1d_0.

The first step is to find out whether the number known fully is of the form k or $k+1$. In our example, $a_1a_0 = d_1d_0 = 10$ and $b_1b_0 = c_1c_0 = 01$. Clearly, since b_1b_0 is less than a_1a_0, it implies that a is of the form $k + 1$. Thus, $4a = 4k + 4$ can be obtained by

merely appending 2 0's to a. Alternatively, $4k$ can be obtained by appending 2 0's to $a-1$.

The second step is to calculate m. This is trivial once we know the number of 1's present in $a_0b_0c_0d_0$. In our example, $m=2$.

The sum $a+b+c+d$ thus becomes $4k+m=4\times21+2=86$ which is as expected. Note that the sum has been computed using 8-bit information from a alone. Only the 2 LSB bits for the other numbers are known.

APPENDIX C

Noise Analysis

C.1 Random noise

Since the received signal is hard-limited in our system, any noise seen at the input is treated in a similar manner. In other words, the AWGN that we expect at the input will be converted to a binary signal, too. This will shape the noise so that the high frequency components are enhanced and the low frequency components are suppressed. This interpretation holds true only while we continue to work with voltages however. A better approach, as proposed in (Sepke *et al.*, 2009), is to interpret the voltage noise at the input as jitter at the output.

This approach is suitable for our analysis because we have mainly focused on the time-delay associated between signals received at different receivers. Therefore, it is apt to convert voltage noise to these terms as well. The aforementioned paper gives us the equation

$$\sigma_{\Delta_i}^2 = \overline{v_n^2} \left| \frac{\mathrm{d}v_i}{\mathrm{d}t} \right|_{v_i = v_{CM}}^{-2}$$

where,

σ_{Δ_i} represents the standard deviation of the jitter observed in Δ_i,

$\overline{v_n^2}$ represents the rms value of the voltage noise seen at the input of the comparator,

v_i is the input signal to the comparator, and

v_{CM} is the common-mode voltage level at the input of the comparator

In our case, we know that $v_i = Asin(\omega t)$ so that its differential then becomes $\frac{\mathrm{d}v_i}{\mathrm{d}t} = A\omega cos(\omega t)$. This, when evaluated at the instants where $v_i = v_{CM}$ which are

essentially the times when the sine wave crosses zero, give us that

$$\left|\frac{\mathrm{d}v_i}{\mathrm{d}t}\right|^2_{v_i=v_{CM}} = A^2\omega^2$$

so that we have

$$\sigma^2_{\Delta_i} = \frac{\overline{v_n^2}}{A^2\omega^2}$$
$$\therefore \sigma^2_{\Delta_i} = \frac{1}{\omega^2 SNR_i}$$

Assuming that the jitter at the output of different comparators is uncorrelated, we have

$$\sigma^2_{\Delta_{ij}} = \frac{1}{\omega^2}\left(\frac{1}{SNR_i} + \frac{1}{SNR_j}\right)$$

We note that increasing the frequency of the input signal reduces the jitter at the output. This is agreeable because we expect that as the time spent near the zero-crossings decreases, the influence of noise should reduce as well. In the limiting case where the input signal itself is a binary signal with infinitely fast transitions, the influence of noise should be negligible. The SNR term being inversely proportional to the output jitter is also as expected because a better SNR is bound to give better performance.

C.2 Quantization Noise

As is true of any representation of analog values in a digital format, because of the "digitization" that we introduce by using the TDC, quantization error is added to the system as well. The quantization in this case depends on the sampling frequency used by the FPGA system to measure the time interval between 2 received signals. Figure C.1 shows the normalized quantization error for different sampling rates.

The figures are obtained as follows:

1. For each point on the plane $z = 2$m, we assume the presence of a single source at that point (x_o, y_o).

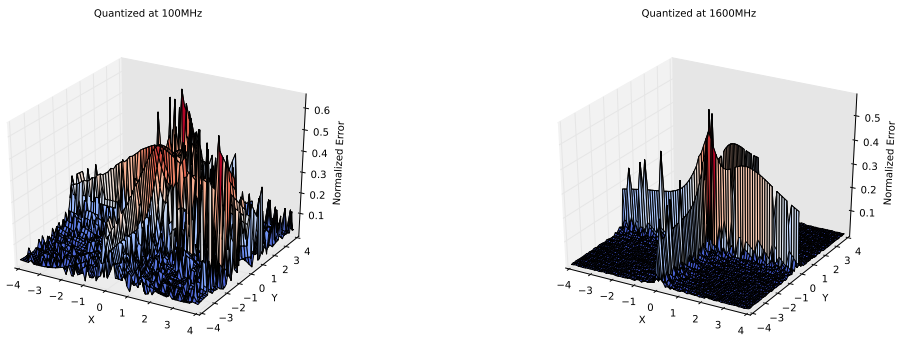


Figure C.1: Quantization error at $z = 2$m

2. The delay between the received signals is then estimated and quantized according to the sampling frequency.

3. Using this value of the observed time delay, the source location is back-calculated using the TDOA equations for x and y given by A.8 and A.7 to give (x', y').

4. The normalized error is then plotted as $e = \dfrac{\sqrt{(x' - x_o)^2 + (y' - y_o)^2}}{\sqrt{x_o^2 + y_o^2 + z^2}\,|_{z=2}}$

The figures clearly show that as the sampling rate is increased, the quantization error reduces drastically. The error still remains high when either of the coordinates become close to 0, but this indicates a theoretical limit governing the equations being used. This motivates our need to increase the sampling rate from the 100MHz clock that the FPGA uses to a rate as high as can be achieved.

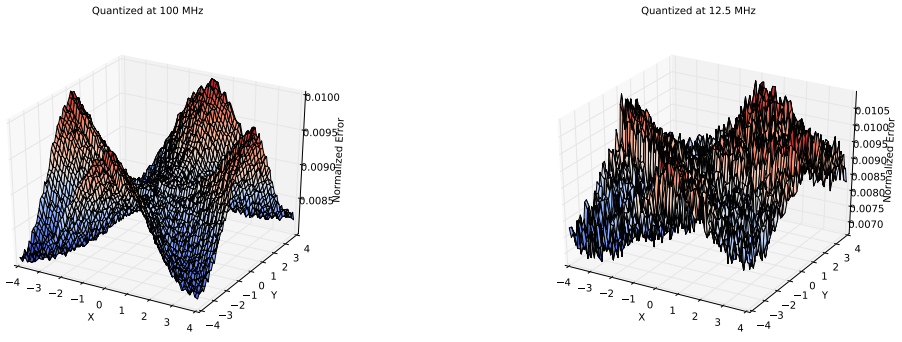


Figure C.2: Quantization error at $z = 2$m assuming spherical localization

Upon using spherical coordinates, the effect of sampling frequency on performance is greatly reduced, as can be seen from Figure C.2. The improvement in robustness can be attributed to the fact that the previous method estimated all 3 parameters whereas this method estimates only the 2 directional parameters.

APPENDIX D

Sensor position calibration

In this section, we consider the reverse problem of calibrating sensor locations in a sensor array. (Qu and Xie, 2012) presents a similar work where they calibrate sensor locations without assuming a known transmitter location. Assuming that knowledge of the transmitter location is available allows us to use a deterministic approach. Consider the situation shown in Figure D.1.

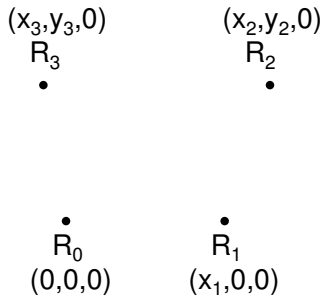


Figure D.1: Unknown sensor locations

Without loss of generality, we can assume sensor R_0 to be the origin of the system and further define the axes in such a way that sensor R_1 has only X as its non-zero coordinate. An implicit assumption being made is that all the sensors are on the same plane and this is defined by $z = 0$. Our goal is to establish locations where transmitters should be placed, so that the unknowns x_1, x_2, x_3, y_2, and y_3 can be estimated by measuring the time difference between the signals received at each of the sensors.

Because of the particular initial arrangement that we have chosen, there are further implicit conditions on some of the x_i's and y_i's. These are $x_1 \geq 0$, $y_2 \geq 0$ and $y_3 \geq 0$.

Recall the equations that we used for TDOA source localization:

$$
\begin{aligned}
x^2 + y^2 + z^2 &= c^2\tau^2 \\
(x - x_1)^2 + y^2 + z^2 &= c^2(\tau + \Delta_{10})^2 \\
(x - x_2)^2 + (y - y_2)^2 + z^2 &= c^2(\tau + \Delta_{20})^2 \\
(x - x_3)^2 + (y - y_3)^2 + z^2 &= c^2(\tau + \Delta_{30})^2
\end{aligned}
$$

Substituting the first transmitter location as $(0, 0, z_0)$ in these equations gives $\tau = \frac{z_0}{c}$. Thus, we have $x_1 = c\sqrt{\Delta_{10,1}\left(\frac{2z_0}{c} + \Delta_{10,1}\right)}$ and the following equations:

$$x_2^2 + y_2^2 + z_0^2 = c^2(\frac{z_0}{c} + \Delta_{20,1})^2$$
$$x_3^2 + y_3^2 + z_0^2 = c^2(\frac{z_0}{c} + \Delta_{30,1})^2$$

Substituting the second transmitter location as $(x_1, 0, z_0)$ in these equations gives $\tau = \frac{z_0}{c} - \Delta_{10,2}$ and

$$(x_1 - x_2)^2 + y_2^2 + z_0^2 = c^2(\frac{z_0}{c} - \Delta_{10,2} + \Delta_{20,2})^2 = c^2(\frac{z_0}{c} + \Delta_{21,2})^2$$
$$(x_1 - x_3)^2 + y_3^2 + z_0^2 = c^2(\frac{z_0}{c} - \Delta_{10,2} + \Delta_{30,2})^2 = c^2(\frac{z_0}{c} + \Delta_{31,2})^2$$

These equations upon elimination of the y terms give the expressions for x_2 and x_3 in terms of the known parameters and x_1 (precomputed) as

$$x_2 = \frac{x_1}{2} + \frac{c^2}{x_1}\left[\left(\frac{z_0}{c} + \Delta_{20,1}\right)^2 - \left(\frac{z_0}{c} + \Delta_{21,2}\right)^2\right]$$
$$x_3 = \frac{x_1}{2} + \frac{c^2}{x_1}\left[\left(\frac{z_0}{c} + \Delta_{30,1}\right)^2 - \left(\frac{z_0}{c} + \Delta_{31,2}\right)^2\right]$$

The remaining unknowns y_2 and y_3 can then be easily expressed in terms of knowns as

$$y_2 = \sqrt{c^2\left(\frac{z_0}{c} + \Delta_{20,1}\right)^2 - z_0^2 - x_2^2}$$
$$y_3 = \sqrt{c^2\left(\frac{z_0}{c} + \Delta_{30,1}\right)^2 - z_0^2 - x_3^2}$$

Thus, the coordinates can be determined using only 2 known locations of the source - one directly above the sensor considered to be the origin, and one directly above the sensor that helps define the X-axis.

REFERENCES

1. **Addeo, E. J.**, **J. D. Robbins**, and **G. Shtirmer** (1994). Sound localization system for teleconferencing using self-steering microphone arrays. US Patent 5,335,011.

2. **Balla, A.**, **M. Beretta**, **P. Ciambrone**, **M. Gatta**, **F. Gonnella**, **L. Iafolla**, **M. Mascolo**, **R. Messi**, **D. Moricciani**, and **D. Riondino** (2012). Low resource FPGA-based time to digital converter. *arXiv preprint arXiv:1206.0679*.

3. **Brandstein, M. S.** and **H. F. Silverman** (1997). A practical methodology for speech source localization with microphone arrays. *Computer Speech & Language*, **11**(2), 91 – 126. URL `http://www.sciencedirect.com/science/article/pii/S0885230896900248`.

4. **Busso, C.**, **S. Hernanz**, **C. Chu**, **S. Kwon**, **S. Lee**, **P. G. Georgiou**, **I. Cohen**, and **S. Narayanan**, Smart room: participant and speaker localization and identification. *In Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 2. IEEE, 2005.

5. **Chen, J. C.**, **K. Yao**, and **R. E. Hudson** (2002). Source localization and beamforming. *Signal Processing Magazine, IEEE*, **19**(2), 30–39.

6. **Dudek, P.**, **S. Szczepanski**, and **J. V. Hatfield** (2000). A high-resolution CMOS time-to-digital converter utilizing a vernier delay line. *Solid-State Circuits, IEEE Journal of*, **35**(2), 240–247.

7. **Fischer, S.** and **K. U. Simmer** (1996). Beamforming microphone arrays for speech acquisition in noisy environments. *Speech communication*, **20**(3), 215–227.

8. **Jansson, J.-P.**, **A. Mantyniemi**, and **J. Kostamovaara** (2006). A CMOS time-to-digital converter with better than 10 ps single-shot precision. *Solid-State Circuits, IEEE Journal of*, **41**(6), 1286–1296.

9. **Jin, S.**, **D. Kim**, **H. S. Kim**, **C. H. Lee**, **J. S. Choi**, and **J. W. Jeon**, Real-time sound source localization system based on FPGA. *In Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on*. IEEE, 2008.

10. **Kalisz, J.** (2004). Review of methods for time interval measurements with picosecond resolution. *Metrologia*, **41**(1), 17.

11. **Kalisz, J.**, **R. Szplet**, **J. Pasierbinski**, and **A. Poniecki** (1997). Field-programmable-gate-array-based time-to-digital converter with 200-ps resolution. *Instrumentation and Measurement, IEEE Transactions on*, **46**(1), 51–55.

12. **Kellermann, W.**, A self-steering digital microphone array. *In Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*. 1991. ISSN 1520-6149.

13. **Khalil, F.**, **J. P. Jullien**, and **A. Gilloire** (1994). Microphone array for sound pickup in teleconference systems. *Journal of the Audio Engineering Society*, **42**(9), 691–700.

14. **Kiyohara, K.**, **Y. Kaneda**, **S. Takahashi**, **H. Nomura**, and **J. Kijima**, A microphone array system for speech recognition. *In Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 1. IEEE, 1997.

15. **Kwon, B.**, **Y. Park**, and **Y. sik Park**, Sound source localization for robot auditory system using the summed GCC method. *In Control, Automation and Systems, 2008. ICCAS 2008. International Conference on*. 2008.

16. **Lédeczi, Á.**, **P. Volgyesi**, **M. Maróti**, **G. Simon**, **G. Balogh**, **A. Nádas**, **B. Kusy**, **S. Dóra**, and **G. Pap**, Multiple simultaneous acoustic source localization in urban terrain. *In Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*. IEEE, 2005.

17. **Lin, Q.**, **E. Jan**, and **J. Flanagan** (1994). Microphone arrays and speaker identification. *Speech and Audio Processing, IEEE Transactions on*, **2**(4), 622–629.

18. **Lorenz, R. G.** and **S. P. Boyd** (2005). Robust minimum variance beamforming. *Signal Processing, IEEE Transactions on*, **53**(5), 1684–1696.

19. **Moore, D. C.** and **I. A. McCowan**, Microphone array speech recognition: Experiments on overlapping speech in meetings. *In Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 5. IEEE, 2003.

20. **Mosher, J.** and **R. Leahy** (1999). Source localization using recursively applied and projected (RAP) MUSIC. *Signal Processing, IEEE Transactions on*, **47**(2), 332–340. ISSN 1053-587X.

21. **Nguyen, D.**, **P. Aarabi**, and **A. Sheikholeslami**, Real-time sound localization using field-programmable gate arrays. *In Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, volume 2. IEEE, 2003.

22. **Qu, X.** and **L. Xie**, Source localization by TDOA with random sensor position errors - Part I: Static sensors. *In Information Fusion (FUSION), 2012 15th International Conference on*. 2012.

23. **Rui, Y.** and **D. Florencio**, New direct approaches to robust sound source localization. *In Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, volume 1. IEEE, 2003.

24. **Schmidt, R.** (1986). Multiple emitter location and signal parameter estimation. *Antennas and Propagation, IEEE Transactions on*, **34**(3), 276–280. ISSN 0018-926X.

25. **Sepke, T.**, **P. Holloway**, **C. G. Sodini**, and **H.-S. Lee** (2009). Noise analysis for comparator-based circuits. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, **56**(3), 541–553.

26. **Stoica, P.**, **P. Handel**, and **A. Nehoral** (1995). Improved sequential MUSIC. *Aerospace and Electronic Systems, IEEE Transactions on*, **31**(4), 1230–1239. ISSN 0018-9251.

27. **Valin, J.-M.**, **F. Michaud**, **J. Rouat**, and **D. Létourneau**, Robust sound source localization using a microphone array on a mobile robot. *In Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2. IEEE, 2003.

28. **Yan, Y.**, **H. Wang**, **X. Shen**, **F. Yang**, and **Z. Chen**, Efficient convex optimization method for underwater passive source localization based on RSS with WSN. *In Signal Processing, Communication and Computing (ICSPCC), 2012 IEEE International Conference on*. IEEE, 2012.

29. **Zimmermann, B.** and **C. Studer**, FPGA-based real-time acoustic camera prototype. *In Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, 2010.

30. **Ziskind, I.** and **M. Wax** (1988). Maximum likelihood localization of multiple sources by alternating projection. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, **36**(10), 1553–1560. ISSN 0096-3518.