REAL TIME IMPLEMENTATION OF AN ACOUSTIC RADAR ON FIELD PROGRAMMABLE GATE ARRAYS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MURAT ARSLAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2012

Approval of the thesis:

REAL TIME IMPLEMENTATION OF AN ACOUSTIC RADAR ON FIELD PROGRAMMABLE GATE ARRAYS

submitted by **MURAT ARSLAN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _______________

Prof. Dr. İsmet Erkmen
Head of Department, **Electrical and Electronics Engineering** _______________

Prof. Dr. Temel Engin Tuncer
Supervisor, **Electrical and Electronics Engineering Department** _______________

Examining Committee Members:

Prof. Dr. Temel Engin Tuncer
Electrical and Electronics Engineering Dept., METU _______________

Prof. Dr. Buyurman Baykal
Electrical and Electronics Engineering Dept., METU _______________

Prof. Dr. Gözde Akar
Electrical and Electronics Engineering Dept., METU _______________

Assoc. Prof. Dr. Çağatay Candan
Electrical and Electronics Engineering Dept., METU _______________

Dr. Güzin Kurnaz, PhD.
HBT SKTM, ASELSAN _______________

Date: _______________

ABSTRACT

REAL TIME IMPLEMENTATION OF AN ACOUSTIC RADAR ON FIELD PROGRAMMABLE GATE ARRAYS

Arslan, Murat
M.S., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. Temel Engin Tuncer

December 2012, 133 pages

Radar is a system which uses electromagnetic waves to determine the range, direction and velocity of objects. In this thesis, an Acoustic Radar system is designed and implemented. Acoustic Radar uses sound waves to localize objects. While such a system has several limitations compared to its electromagnetic counterpart, the simplicity and low cost of its front-end makes it a good testbed for realizing the functions of a radar system. In this thesis, a six element transmit-receive array is constructed using loudspeakers and microphones. Six channel A/D and D/A converter boards are integrated with an FPGA unit where all the processing for transmit and receive functions are implemented. Signal processing is composed of filtering, beamforming, cross correlation and decision processes. The hardware platform is controlled by a computer through an ethernet connection. A user interface is provided where the radar functions can be controlled and changed. The tests for the acoustic radar are performed in an uncontrolled laboratory environment where there are several objects which generate multipaths. It is shown that the system works consistently and can localize objects accurately.

Keywords: acoustic radar, acoustic imaging, phased arrays, delay-and-sum beamformer

ÖZ

ALANDA PROGRAMLANABİLİR KAPI DİZİLERİNDE GERÇEK ZAMANLI BİR AKUSTİK RADAR UYGULAMASI

Arslan, Murat
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Temel Engin Tuncer

Aralık 2012, 133 sayfa

Radar; nesnelerin uzaklıklarını, yönlerini ve hızlarını belirlemek için elektromanyetik dalgalar kullanan bir sistemdir. Bu tez çalışmasında akustik bir radar sistemi tasarlanmış ve gerçeklenmiştir. Akustik Radar nesnelerin yerlerini belirlemek için ses dalgaları kullanır. Böyle bir sistem, elektromanyetik emsali ile karşılaştırıldığında birçok sınırlamaya sahip olmasına rağmen basit ve düşük maliyetli ön ucu onu radar fonksiyonlarının gerçeklenmesi için uygun bir sınama ortamı haline getirmektedir. Bu tez çalışmasında altı elemanlı bir alıcı-verici dizisi hoparlör ve mikrofonlar kullanılarak oluşturulmuştur. Altı kanallı A/D ve D/A çevirici devre kartları bütün gönderme ve alma işaret işleme fonksiyonlarının gerçeklendiği bir FPGA birimi ile entegre edilmiştir. İşaret işleme fonksiyonları süzme, hüzmeleme, çapraz korelasyon ve karar verme işlemlerinden oluşur. Donanım platformu bir bilgisayar tarafından ethernet bağlantısı aracılığıyla kontrol edilmektedir. Radar fonksiyonlarının kontrol edilebileceği ve değiştirilebileceği bir kullanıcı arayüzü sunulmaktadır. Akustik radar testleri çok-yolluluk etkileri yaratan birçok nesnenin yer aldığı kontrolsüz bir laboratuvar ortamında yürütülmüştür. Sistemin tutarlı bir biçimde çalıştığı ve nesnelerin yerlerini doğru olarak tespit edebildiği ispatlanmıştır.

Anahtar Kelimeler: akustik radar, akustik görüntüleme, fazlı diziler, geciktir-ve-topla hüzme şekillendirici

To my family

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my supervisor Prof. Dr. Temel Engin Tuncer for his precious support, ideas and trust throughout my thesis work. Without his everlasting patience, excellent guidance, motivating attitude and practical approach to the problem, I could not complete this thesis work.

I would like to thank my colleague Soner Yeşil for his support, motivating discussions, precious ideas and suggestions on the implementation of the system. He has obvious effects on my fundamental decisions throughout the thesis study.

I would like to thank my colleague Barış Bölk̈başı for helping me in designing the PCB which connects two fundamental hardware units in the system and for always having time for my questions about hardware issues of the system.

I would like to thank Barış Tanyeri (accounts manager in Texas Instruments Inc., Turkey) and Texas Instruments Inc. for donating one of the main hardware units of the system, which is PCM1602 DAC Evaluation Module.

I would like to thank my colleague Ersin Karatop for taking time and helping me in cabling, soldering and other hardware modifications despite his heavy work load.

I would like to thank Aselsan Inc. for supporting my graduate study. I would like to thank my close friends Hakan Erkal, Önder Bilgiç and all the others whom I cannot state individually here for their motivating support.

I would like to thank the METU SAM (Sensor Array and Multichannel Signal Processing) Research Laboratory People Yılmaz Kalkan and Ahmet Musab Elbir for their good company and support.

Last but not the least; I would like to thank my family. Not only this thesis work, nothing would be possible without them.

TABLE OF CONTENTS

ABSTRACT iv

ÖZ v

ACKNOWLEDGMENTS viii

TABLE OF CONTENTS ix

LIST OF TABLES xii

LIST OF FIGURES xiii

CHAPTERS

1 INTRODUCTION 1

2 BACKGROUND 3

2.1 Array Signal Processing 3

2.1.1 Spatial Signals 4

2.1.2 Near and Far Field Concept 5

2.1.3 Array Model Representation 7

2.2 Array Beamforming 13

2.2.1 Delay and Sum Technique 14

2.2.2 Designing the Maximum SNR Spatial Filter [15] 19

3 ACOUSTIC RADAR 22

3.1 System Structure 22

3.1.1 Radar System Controller 23

3.1.2 Transmitter 24

3.1.3 Receiver 25

3.1.4 System Configuration and Display Unit 25

3.1.5 Memory Unit 26

3.1.6 Transmit and Receive Arrays 27

3.1.7 Acoustic Radar System Parameters 27

3.1.7.1 The Radar Equation 28

3.1.7.2 Pulse Width 29

3.1.7.3 Wavelength 30

3.1.7.4 Pulse Repetition Period 30

3.1.7.5 Pulse Shape 31

4 FPGA IMPLEMENTATION OF RADAR PROCESSING 34

4.1 Hardware Description 35

4.1.1 DIGILENT ATLYS Spartan-6 FPGA Board 35

4.1.2 TI PCM1602 DAC Evaluation Module 37

4.1.3 TI ADS8364 ADC Evaluation Module 39

4.2 Description of FPGA Building Blocks 40

4.2.1 Transmit Pulse Generator Module 40

4.2.2 PCM1602 DAC Controller Module 43

4.2.3 ADS8364 ADC Controller Module 45

4.2.4 DC Offset Remover 47

4.2.5 6-Channel FIR Low-Pass Filter Module 50

4.2.6 Receive Delay and Sum Beamformer Module 54

4.2.7 Correlator Module 57

4.2.8 Clock Generation Module 61

4.2.9 Ethernet Controller Module 62

4.2.10 Acoustic Radar System Controller Module 65

4.2.10.1 System Registers 66

4.2.10.2 Description of System Operation 68

5 EXPERIMENTAL RESULTS 77

5.1 PC Controlled Transmit Beamformer 77

5.1.1 5.1.1 Description of the System Parameters 78

5.1.2 Description of System Operation 79

5.1.2.1 UART_TO_PULSE_GEN_FSM MODULE . . 79

5.1.2.2 UART_CTRL 81

5.1.3 Results 82
5.2 Acoustic Radar Experiments 88
5.2.1 Experiment 1: ADC Capture over UDP 88
5.2.2 Experiment 2: Acoustic Radar System Evaluation 91
6 CONCLUSIONS 115
REFERENCES 117
APPENDICES
A MODULE PORT DESCRIPTIONS 119
A.1 TRANSMIT PULSE GENERATOR PORT DESCRIPTIONS 119
A.2 PCM1602 DAC CONTROLLER PORT DESCRIPTIONS 120
A.3 ADS8364 ADC CONTROLLER PORT DESCRIPTIONS 122
A.4 DC OFFSET REMOVER PORT DESCRIPTIONS 124
A.5 6-CHANNEL FIR LOW-PASS FILTER PORT DESCRIPTIONS . . 125
A.6 RECEIVE DELAY AND SUM BEAMFORMER PORT DESCRIPTIONS 125
A.7 CORRELATOR PORT DESCRIPTIONS 128
A.8 CLOCK GENERATOR PORT DESCRIPTIONS 130
A.9 ETHERNET CONTROLLER PORT DESCRIPTIONS 131

LIST OF TABLES

TABLES

Table 4.1 Implemented Low-Pass Filter Specifications 50

Table A.1 Transmit Pulse Generator Module Port Description 120

Table A.2 PCM1602 DAC Controller Port Description 122

Table A.3 ADS8364 ADC Controller Port Description 124

Table A.4 DC Offset Remover Port Description 125

Table A.5 6-Channel FIR Low-Pass Filter Module Port Description 125

Table A.6 Receive Delay and Sum Beamformer Port Description 128

Table A.7 Correlator Port Description 130

Table A.8 Clock Generator Port Description 131

Table A.9 Ethernet Controller Port Description 133

LIST OF FIGURES

FIGURES

Figure 2.1 Single antenna vs. array of antennas 4

Figure 2.2 There dimensional space in spherical coordinates 5

Figure 2.3 A sensor array receiving the transmitted signal from a point source 6

Figure 2.4 Energy spectrum of a bandpass signal 8

Figure 2.5 Lowpass equivalent of the bandpass signal whose spectrum is given in Figure 2.4 9

Figure 2.6 ULA Delay-and-Sum Beamforming 17

Figure 2.7 Directivity response of the Acoustic Radar Beamformer with steer angle chosen as 90 ° (Same for transmit and receive arrays) 18

Figure 3.1 Acoustic Radar System Structure 22

Figure 3.2 Phased Array Transmit Steer 24

Figure 3.3 Acoustic Radar Display Unit 26

Figure 3.4 The Acoustic Radar Pulse 31

Figure 3.5 Acoustic Radar Pulse Magnitude Response 32

Figure 3.6 Ambiguity function for the Acoustic Radar Pulse 33

Figure 3.7 Ambiguity function for an unmodulated (left) pulse and a chirp pulse (rigth) of the same length as the Acoustic Radar Pulse 33

Figure 4.1 Implemented Radar System Structure 34

Figure 4.2 Top view and functional block diagram of ATLYS FPGA Board 35

Figure 4.3 PCB for connecting ATLYS Board to ADS8364 ADC EVM 37

Figure 4.4 PCM1602 DAC Evaluation Module Functional Block Diagram 38

Figure 4.5 Top view of ADS8364 Evaluation Module 39

Figure 4.6 Tranmit Pulse Generator Functional Block Diagram 41

Figure 4.7 Delay Lookup Table Content 42

Figure 4.8 PCM1602 DAC Controller Functional Block Diagram 44

Figure 4.9 Standard Audio Data Input Formats 45

Figure 4.10 ADS8364 Data Capture Timing Diagram 46

Figure 4.11 ADS8364 Data Capture Timing Diagram 47

Figure 4.12 Analog RC Circuit to Determine the DC Offset of a Signal 47

Figure 4.13 DC offset of a sinusoidal signal calculated by the RC Circuit 49

Figure 4.14 *DC Offset Remover*, a digital implementation of the RC Circuit 49

Figure 4.15 Frequency Response of the Implemented Low-Pass Filter 50

Figure 4.16 A direct form realization of an FIR filter (1 channel) 51

Figure 4.17 Pipelined structure of the FIR filter to improve timing (1 channel) 52

Figure 4.18 6-channel low-pass FIR filter structure 52

Figure 4.19 DSP48A1 Slice 53

Figure 4.20 DSP48A1 Slice, used parts highlighted 54

Figure 4.21 Receiver Delay-and-Sum Beamformer Functional Block Diagram 55

Figure 4.22 Receiver Delay-and-Sum Beamformer Finite State Machine Flow Diagram 56

Figure 4.23 Multiply-Accumulate circuit which used to calculate a correlation sample . 58

Figure 4.24 Correlator Functional Block Diagram 60

Figure 4.25 Correlator Finite State Machine Flow Diagram 60

Figure 4.26 Clock Generation Module Functional Block Diagram 61

Figure 4.27 Ethernet Controller Module Functional Block Diagram 63

Figure 4.28 User Datagram Protocol Packet Format 64

Figure 4.29 Acoustic Radar System Controller Functional Diagram 66

Figure 4.30 Acoustic Radar System Controller Finite State Machine Parts 68

Figure 4.31 Acoustic Radar State Machine Part I: Observation Phase 70

Figure 4.32 Acoustic Radar State Machine Part II: Transmit-Receive Phase 71

Figure 4.33 Acoustic Radar State Machine Part III: Data Feed Phase 74

Figure 5.1 PC Controlled Transmit Beamformer Functional Block Diagram 78

Figure 5.2 PC Controlled Transmit Beamformer Finite State Machine Flow Diagram . 80

Figure 5.3 UART Controller Module Functional Block Diagram 81

Figure 5.4 PC Controlled Transmit Beamformer Test Setup 82

Figure 5.5 *Transmit Pulse Generator* output, *steer angle* is 80° 84

Figure 5.6 *Transmit Pulse Generator* output, *steer angle* is 170° 85

Figure 5.7 Transmit Pulse Generator output in *infinite mode* for three different *pulse repetition parameter* values 87

Figure 5.8 ADC Capture over UDP Functional Block Diagram 89

Figure 5.9 Channel-0 Captured Echo Comparison, Raw vs DC Offset Removed, Low-Pass Filtered Waveform 90

Figure 5.10 6-Channel Capture, duration: 58.8 msec (1 PRF), target position: 3 m, 90° 91

Figure 5.11 Target placement for Scenario 1 92

Figure 5.12 Target placement for Scenario 2 92

Figure 5.13 Transmitted CH4 and CH5 waveforms from the receive array's point of view, no phase difference observed, target position: 3.5 m, 90° 94

Figure 5.14 Transmitted waveforms for all channels from the receive array's point of view, no phase difference observed, target position: 3.5 m, 90° 94

Figure 5.15 Received echo from CH4 and CH5, no phase difference observed, target position: 3.5 m, 90° 95

Figure 5.16 Received echo from all channels, no phase difference observed, target position: 3.5 m, 90° 96

Figure 5.17 Entire transmit-receive cycle captured, target position: 3.5 m, 90° 97

Figure 5.18 Beamformer Output, number of captures: 1, target position: 3.5 m, 90° . . 98

Figure 5.19 Beamformer Output for different values of *number of captures per angle* parameter, target position: 3.5 m, 90° 99

Figure 5.20 Normalized beamformer outputs zoomed, *number of captures* parameter varied, target position: 3.5 m, 90° 99

Figure 5.21 Correlator Output, decimation factor: 1, target position: 3.5 m, 90° 100

Figure 5.22 Correlator Output, decimation factor: 2, target position: 3.5 m, 90° 101

Figure 5.23 Correlator Output, decimation factor: 32, target position: 3.5 m, 90° 102

Figure 5.24 Correlator Output for different values of *correlator decimation factor* parameter, target position: 3.5 m, 90° 102

Figure 5.25 Radar screenshot, target position: 3.5 m, 90° 103

Figure 5.26 Transmitted CH1 and CH2 waveforms from the receive array's point of view, note the phase difference between channels, target position: 2 m, 60° 104

Figure 5.27 Transmitted waveforms for all channels from the receive array's point of view, note the phase difference between channels, target position: 3.5 m, 90° . . . 105

Figure 5.28 Received echo from CH1 and CH2, note the phase difference, target position: 2 m, 60° 106

Figure 5.29 Received echo from all channels, no phase difference observed, target position: 2 m, 60° 107

Figure 5.30 Entire transmit-receive cycle captured, target position: 2 m, 60° 108

Figure 5.31 Beamformer Output, number of captures: 1, target position: 2 m, 60° . . . 109

Figure 5.32 Beamformer Output for different values of *number of captures per angle* parameter, target position: 2 m, 60° 110

Figure 5.33 Normalized beamformer outputs zoomed, *number of captures* parameter varied, target position: 2 m, 60° 110

Figure 5.34 Correlator Output, decimation factor: 1, target position: 2 m, 60° 111

Figure 5.35 Correlator Output, decimation factor: 2, target position: 2 m, 60° 112

Figure 5.36 Correlator Output, decimation factor: 32, target position: 2 m, 60° 112

Figure 5.37 Correlator Output for different values of *correlator decimation factor* parameter, target position: 2 m, 60° 113

Figure 5.38 Radar screenshot, target position: 2 m, 60° 114

CHAPTER 1

INTRODUCTION

Phased array systems have a wide range of usage in fields such as radar, sonar, tomography and cellular communications [1]. The phased array systems mostly use antennas which operate with radio frequency waves, requiring expensive and technologically advanced equipment. A smaller and cheaper system may be designed and implemented using acoustic waves rather than electromagnetic waves, which may exactly illustrate the concepts of a phased array device. So, this work is concentrated on designing and implementing an acoustic phased array radar system. Together with the fact that it provides a simpler and cheaper test bed to examine the phased array radar behaviors, this work has also a significance in that phased array principles have not been extensively applied for acoustic imaging in air.

The history of acoustic radar applications goes back to the Acoustic Locators used in World War I [2]. They were used for passive detection of aircraft by picking up the noise of the engines. Upon the introduction of radar before and during World War II, which was obviously a more effective technique, those passive acoustic locators have become obsolete. But, there is still a wide application area of acoustic waves in air. In [3], a theoretical discussion about an atmospheric acoustical application, called SODAR, is given. SODAR (sonic detection and ranging) is a RADAR like system which, however, uses sound waves rather than radio waves to remotely measure the vertical turbulence structure and the wind profile of the lower layer of the atmosphere. Phased-array sodar systems, in which the beams are electronically steered by phasing the transducers appropriately as in our particular application, were developed in the United States during late 1980‘s and early 1990‘s by Xonics, Radian Corporation and AeroVironment. Another example for the acoustic radar applications is the travel aid devices which are used to improve the mobility of the visually impaired people [4]. The devices in [5] , [6] are simplest examples as they do not scan the environment although they are both based on array structure. As mentioned earlier, phased array principles have not been extensively used in air acoustical applications. In [7], an ultrasonic obstacle detector which uses the phased array principles is introduced. This application is still not a complete one as it only performs receive beamforming using a microphone array and has a single transducer for transmission. A more complete acoustic imaging application is given in [8], which is composed of two separate linear transmit and receive arrays. Transmit and receive arrays both use the phased beamforming technique to electronically sweep the acoustic beam.

In this thesis work, an acoustic radar system is implemented, which has separate transmit and receive arrays composed of six elements. Unlike [8] which employs ultrasound beams, the particular application uses acoustic sound waves which are generated by a 6 element loud-speaker transmit array. A 6 element microphone receive array is used for receive functioning. All signal processing algorithms are implemented in real-time on FPGA which allows fast operation with a 100 MHz clock frequency. The System performs a 2-dimensional scan from $0°$ to $180°$ with a resolution of $10°$. The system is tested and verified under various scenarios. In CHAPTER 2, a background information is given, which is mainly based on array signal processing and beamforming. In CHAPTER 3, a conceptual structure of the system is given, the theoretical building blocks are described briefly. In CHAPTER 4, a detailed description of the implementation is given, the functionality and design details of each FPGA building block is explained. In CHAPTER 5, the implementation results, test and verification details are given. Chapter 6 is the conclusions part.

CHAPTER 2

BACKGROUND

In this chapter, the theoretical background which constitutes the base of the implemented acoustic radar system is given.

2.1 Array Signal Processing

Sensor arrays are used to spatially sample the space and extract both temporal and spatial parameters of signals. For a system which utilizes temporal information only, filter structures which are sensitive to certain frequencies are used. However, for a system which also needs spatial information, a direction sensitivity rather than frequency selection is needed. At this point, the concept of spatial filtering comes into the scene which is analoguous to the temporal filtering in the sense that in spatial filtering, the filter is sensitive to the direction, i.e., passes signals from certain directions while rejecting those from other directions, while in temporal filtering, the discrimination is in the frequency content of the incoming signal. The spatial discrimination may be accomplished by a single sensor or an array of sensors as given in Figure 2.1. The spatial discrimination ability of a single sensor is dependent on the geometrical structure of the sensor. Also, in order to change the directivity pattern, i.e. the look direction of the sensor, a mechanical intervention must be applied. An array of sensors overcomes these disadvantages of a single sensor. The directivity pattern and look direction of a sensor array is not that dependent to the physical characteristics and may be altered by changing the phase delay in individual channels rather than moving the sensor array mechanically, which is a big advantage over the single sensor case.

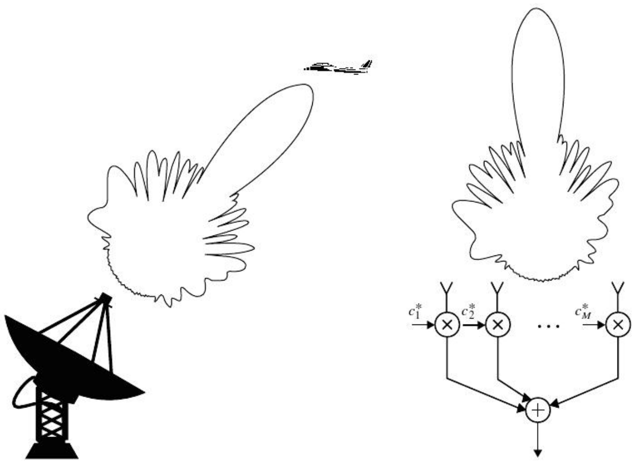


Figure 2.1: Single antenna vs. array of antennas

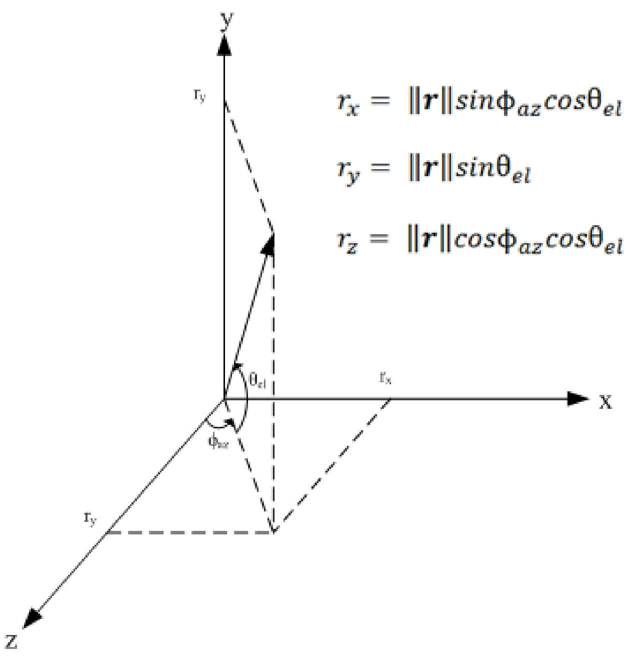
2.1.1 Spatial Signals

Spatial signals are waves propagating through space whose charasteristics are defined by the wave equations. The spatial signal at a particular point specified by the vector r may be represented either in Cartesian coordinates (x, y, z) or in spherical coordinates (R, ϕ_{az}, θ_{el}) as in Figure 2.2, where R is the distance of wave from the origin, ϕ_{az} is the azimuth angle and θ_{el} is the elevation angle.

The spatial wave which is transmitted from a source located at $\mathbf{r}_0$ may be represented as follows;

$$s(t,\mathbf{r}) = \frac{A}{||\mathbf{r}-\mathbf{r}_0||}e^{j2\pi f_c(t-\frac{||\mathbf{r}-\mathbf{r}_0||}{c})} \tag{2.1}$$

where A is the complex amplitude, f_c is the carrier frequency of the wave, and c is the speed of propagation of the wave, which is speed of sound in our case.



y
x
z
r_y
r_x
r_y
θ_{el}
ϕ_{az}
$r_x = \|\mathbf{r}\| sin\phi_{az} cos\theta_{el}$
$r_y = \|\mathbf{r}\| sin\theta_{el}$
$r_z = \|\mathbf{r}\| cos\phi_{az} cos\theta_{el}$

Figure 2.2: There dimensional space in spherical coordinates

2.1.2 Near and Far Field Concept

In array signal processing, far field approximation is important which simplifies the computations considerably. It relies on the assumption that if the transmitter is “very“ far away from the receiving array, and the array aperture is “relatively“ small, then the spherical wave front generated by the transmitter can be approximated as a plane wave.

In literature, usually, 3 regions are defined for the space surrounding an antenna, which are, reactive near field, radiating near field (Fresnel) and far-field (Fraunhofer) regions [9]. In the near field, the wave fronts are modeled as spherical. But, in the far field (or Fraunhofer) region, it is plausible to assume that the wave fronts are plane waves.

The far field assumption may be analysed as follows (consider the geometry given in Figure 2.3). δR is the distance between the spherical wavefront and the receiving array plane, which is

$$\delta R = \sqrt{R^2 + \left(\frac{D}{2}\right)^2} - R \tag{2.2}$$

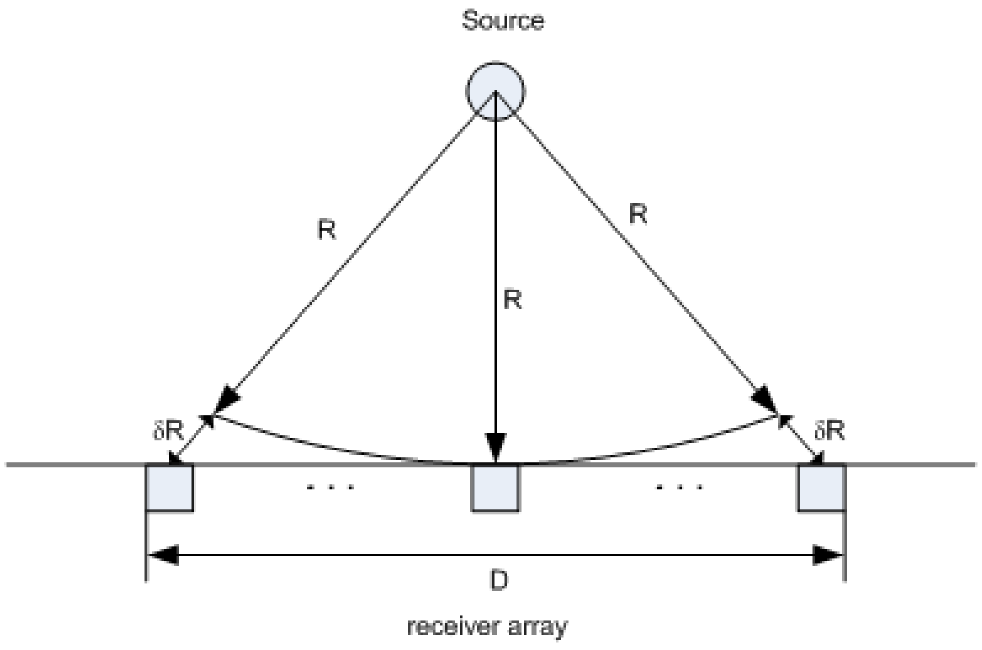


Figure 2.3: A sensor array receiving the transmitted signal from a point source

where R is the distance between the receiving array and the point transmitter and D is the array aperture. Note that, in the far field, D≪R, (2.2) may be approximated using the binomial expansion $\left((x+1)^{\frac{1}{2}} = 1 + \frac{x}{2} - \frac{x^2}{8} ...\right)$ such that,

$$\delta R = \left(\sqrt{1 + \left(\frac{D}{2R}\right)^2} - 1\right) R = \left[1 + \frac{\left(\frac{D}{2R}\right)^2}{2} - 1\right] R = \frac{D^2}{8R} \tag{2.3}$$

In literature [10], an array is assumed to be in the far-field if the distance δR is less than one sixteenth of the wavelength. Therefore, using the condition of $\delta R \leq \frac{\lambda}{16}$, (2.3) reduces to

$$\delta R = \frac{D^2}{8R} \leq \frac{\lambda}{16} \rightarrow R \geq \frac{2D^2}{\lambda} \tag{2.4}$$

(2.4) may further be simplified if the receiving array is an Uniform Linear Array (ULA) with L elements and the inter-element spacing is chosen as $d = \frac{\lambda}{2}$ (which is the maximum distance satisfying the spatial sampling theorem, in other words, in order to prevent the spatial aliasing

[11]) such that

$$
\begin{aligned}
D = d\,(L-1) &= \left(\frac{\lambda}{2}\right)(L-1) \\
R &\geq \left(\frac{2}{\lambda}\right)\left(\left(\frac{\lambda}{2}\right)(L-1)\right)^2 \\
R &\geq \frac{\lambda\,(L-1)^2}{2}
\end{aligned}
\tag{2.5}
$$

In our case, the wavelength $\lambda = c/f = 346.13/1417 = 0.2443$ and number of array elements $L = 6$. So, the condition for far-field approximation in the particular application is $R \geq 3.0538\ m$.

2.1.3 Array Model Representation

In this part, a general array model will be established [12]. First, *single source* case will be considered, then, general array model for the multiple source case will be developed using superposition principle.

Consider the single source case in which only one waveform impinges upon the array. Let x(t) is the value of that source signal at some *reference point* at time *t*. The output of sensor k can be written as

$$\bar{y}_k(t) = \bar{h}_k(t) * x(t - \tau_k) + \bar{e}_k(t) \tag{2.6}$$

where τ_k is the time needed for the wave to travel from the reference point to sensor k, $\bar{h}_k(t)$ is the impulse response of the kth sensor, "*" is the convolution operation and $\bar{e}_k(t)$ is the *additive noise*. In (2.6), $\bar{h}_k(t)$ is assumed to be known. However, x(t) and τ_k are unknown. The parameter which is intended to be estimated is τ_k, because it has the information related to the location of the source. So, as the source signal x(t) is unknown, the problem is nothing but a *time delay estimation for the unknown input case*.

Equation (2.6) can be simplified significantly if the signals are assumed to be *narrowband*. Let X(ω) be the fourier transform of x(t),

$$X(\omega) = \int_{-\infty}^{+\infty} x(t) e^{-i\omega t} dt \qquad (2.7)$$

Similarly, let $\overline{H}_k(\omega)$, $\overline{Y}_k(\omega)$ and $\overline{E}_k(\omega)$ be the fourier transforms of signals $\overline{h}_k(t)$, $\overline{y}_k(t)$ and $\overline{e}_k(t)$, respectively. So, in frequency domain, (2.6) is

$$\overline{Y}_k(\omega) = \overline{H}_k(\omega) X(\omega) e^{-i\omega\tau_k} + \overline{E}_k(\omega) \qquad (2.8)$$

Generally in real systems, x(t) is a *bandpass signal*. It is obvious from Figure 2.4 that a bandpass is completely defined by its corresponging *lowpass signal* s(t). The process of obtaining x(t) from s(t) is called *modulation* and the reverse process of obtaining s(t) from x(t) is called *demodulation*.

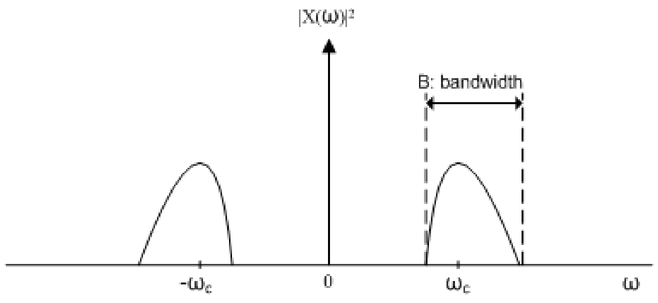


Figure 2.4: Energy spectrum of a bandpass signal

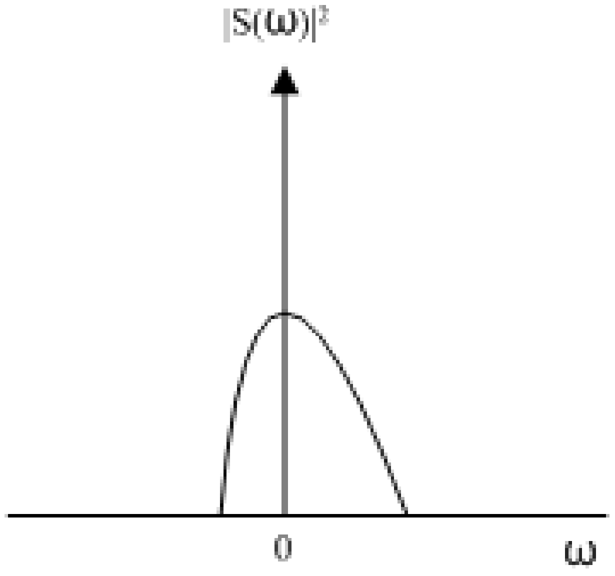


Figure 2.5: Lowpass equivalent of the bandpass signal whose spectrum is given in Figure 2.4

<u>Modulation Process</u>: The bandpass signal x(t) is obtained from the baseband signal s(t) via *modulation* process as mentioned above. Multiplying s(t) with $e^{i\omega_c t}$ results in a shift of ω_c in the spectrum, as expressed in (2.9).

$$\int_{-\infty}^{\infty} s(t)e^{i\omega_c t}e^{-i\omega t}d\omega = \int_{-\infty}^{\infty} s(t)e^{-i(\omega-\omega_c)}d\omega = S(\omega-\omega_c) \tag{2.9}$$

If x(t), which is obtained from s(t) via modulation process, is a real valued signal, it must have an even spectrum. In such a case, in addition to the frequency shift ω_c to the right, there is also a translation to the left of the folded and complex-conjugated baseband spectrum. So, the following expression for the modulated spectrum $X(\omega)$ is obtained;

$$X(\omega) = S(\omega-\omega_c) + S^*(-(\omega+\omega_c)) \tag{2.10}$$

If (2.10) is transformed to the time domain, the following result is obtained:

$$\begin{aligned} x(t) &= \frac{1}{2\pi}\int_{-\infty}^{\infty}\left[S(\omega-\omega_c)+S^*(-\omega-\omega_c)\right]e^{i\omega t}d\omega \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty}S(\omega-\omega_c)e^{i(\omega-\omega_c)t}e^{i\omega_c t}d\omega+\left[\frac{1}{2\pi}\int_{-\infty}^{\infty}S(-\omega-\omega_c)e^{-i(\omega+\omega_c)t}e^{i\omega_c t}d\omega\right]^* \\ &= s(t)e^{i\omega_c t}+\left[s(t)e^{i\omega_c t}\right]^* \end{aligned} \tag{2.11}$$

which gives

$$x(t) = 2Re[s(t)e^{i\omega_c t}] \tag{2.12}$$

or

$$x(t) = 2\alpha(t)cos(\omega_c t+\varphi(t)) \tag{2.13}$$

where $\alpha(t)$ and $\varphi(t)$ are the amplitude and phase of s(t), respectively;

$$s(t) = \alpha(t)e^{i\varphi(t)} \tag{2.14}$$

(2.12) can also be written in terms of real (s$_I$(t)) and imaginary (s$_Q(t)$) parts of s(t) as following;

$$x(t) = 2[s_I(t)cos(\omega_c t) - s_Q(t)sin(\omega_c t)] \tag{2.15}$$

Demodulation Process: s(t) is obtained from x(t) via demodulation process. First, the spectrum of x(t), which is X(ω) is translated left by ω_c. Then, the resulting spectrum is low-pass filtered with a low-pass filter whose bandwidth is matched to that of S(ω). After the first step, the spectrum takes the form

$$[S(\omega)+S^*(-\omega-2\omega_c)] \tag{2.16}$$

If this signal is low-pass filtered, the high frequency component vanishes and the resulting spectrum is nothing but that of the baseband signal, which is S(ω).

Now, let us return to our ultimate purpose of obtaining signal model. Consider (2.8), if X(ω) given in (2.10) is inserted in (2.8);

$$\overline{Y}_k(\omega) = \overline{H}_k(\omega)[S(\omega - \omega_c) + S^*(-\omega - \omega_c)]e^{i\omega\tau_k} + \overline{E}_k(\omega) \tag{2.17}$$

Let $\widetilde{y}_k(t)$ be the demodulated signal, i.e.

$$\widetilde{y}_k(t) = \overline{y}_k(t)e^{-i\omega_c t} \tag{2.18}$$

The Fourier transform of $\widetilde{y}_k(t)$ can be expresses as:

$$\widetilde{Y}_k(\omega) = \overline{H}_k(\omega + \omega_c)[S(\omega) + S^*(-\omega - 2\omega_c)]e^{-i(\omega+\omega_c)\tau_k} + \overline{E}_k(\omega + \omega_c) \tag{2.19}$$

When $\widetilde{y}_k(t)$ is low-pass filtered, the high frequency component in (2.19), which is $S^*(-\omega - 2\omega_c)$ is eliminated and the resulting signal is

$$Y_k(\omega) = H_k(\omega + \omega_c)S(\omega)e^{-i(\omega+\omega_c)\tau_k} + E_k(\omega + \omega_c) \tag{2.20}$$

If the received signals are *narrowband*, i.e. $|S(\omega)|$ decreases rapidly with increasing $|\omega|$, (2.20) can be approximated as

$$Y_k(\omega) = H_k(\omega_c)S(\omega)e^{-i\omega_c\tau_k} + E_k(\omega + \omega_c)\ for\ \omega \in \Omega \tag{2.21}$$

where Ω is the lowpass filter's passband. Further assuming that the frequency response of the sensor is flat over the passband Ω, the time domain counterpart of 2.21 would be the following:

$$y_k(t) = H_k(\omega_c)e^{-i\omega_c\tau_k}s(t) + e_k(t) \tag{2.22}$$

where $y_k(t)$ and $e_k(t)$ are the inverse Fourier transmforms of the terms $Y_k(\omega)$ and $E_k(\omega + \omega_c)$, respectively. Now, let us introduce the *array transfer vector*:

$$a(\theta) = \left[H_1(\omega_c)e^{-i\omega_c\tau_1}...H_m(\omega_c)e^{-i\omega_c\tau_m}\right] \tag{2.23}$$

By making use of (2.23), (2.22) can be rewritten as

$$\begin{aligned} y(t) &= a(\theta)s(t) + e(t) \\ &\quad \textit{where} \\ y(t) &= [y_1(t)...y_m(t)]^T \\ e(t) &= [e_1(t)...e_m(t)]^T \end{aligned} \tag{2.24}$$

The dependency of the *array transfer vector* on sensor responses, which are shown as $H_i(\omega_c)$ may be ignored if the sensor responses are independent of the direction of arrival. So, with this assumption, (2.23) reduces to

$$a(\theta) = \left[1\ e^{-i\omega_c\tau_2}...e^{-i\omega_c\tau_m}\right] \tag{2.25}$$

Note that the array model given in (2.24) is for single channel. Extention of this model to multiple source case is straightforward. Using superposition principle, the following model is obtained,

$$\begin{aligned} y(t) = [a(\theta_1)...a(\theta_n)] \begin{bmatrix} s_1(t) \\ ... \\ s_n(t) \end{bmatrix} + e(t) &\triangleq As(t) + e(t) \\ \theta_k &= the\ DOA\ of\ the\ k^{th}\ source \\ s_k(t) &= the\ signal\ corresponding\ to\ the\ k^{th}\ source \end{aligned} \tag{2.26}$$

Note that the array model in (2.26) is developed for *narrowband* signals. But, as explained in Section 3.1.7.5, for our particular acoustic radar application, the narrowband assumption does not hold. In other words, (2.20) cannot be approximated by (2.21) in our case. So, wideband methods must be used. Wideband signals can be decomposed into many narrowband signals using filter banks or the discrete time Fourier transform along temporal domain, picking the frequencies with the highest power. After this decomposition, narrowbands methods can be applied to each of these narrowband components. The disadvantage of this kind of approach is not taking full advantage of the signal's frequency band. Because, some frequency bins which might have information about the direction of arrival is discarded. As well as frequency domain approaches, some time domain methods can be used to deal with wideband signals. *Delay and Sum Beamforming*, which is used in the acoustic radar system, is one of these methods.

2.2 Array Beamforming

A beamformer may be defined as a spatial filter which discriminates signals depending on the direction of arrival in such a way that it maximizes the SNR in the direction of interest [13]. A beamformer may be used both for reception and transmission (beam steering, [14]) of signals, as in the particular acoustic radar implementation. Let's rewrite the signal model developed in the previous section here for a receiving array of L elements;

$$\begin{aligned} y_i(t) &= \alpha_i s(t - T_0 - \tau_i) + e_i(t) \\ &= x_i(t) + e_i(t) \\ where\ i &= 1, 2, ..., L \end{aligned} \tag{2.27}$$

Where α_i is the attenuation factor due to propagation $\alpha_i \in [0,\ 1]$, s(t) is the unknown signal, which may be narrowband or wideband, T_0 is the time required for the signal to propagate from the source to the 1^{th} array element which is the reference sensor, e$_i$(t) is the additive noise at the ith sensor, τ_i is the relative delay between the reference sensor and ith sensor.

From now on, it will be assumed that TDOA (Time Difference of Arrival), which is nothing but the time required for the signal to travel between adjacent array elements, is known (which is actually the case for the particular acoustic radar implementation; because, the direction of arrival angle of the incoming or expected echo is known apriori). Also, it is assumed that channel noise signals are uncorrelated.

2.2.1 Delay and Sum Technique

A simple beamformer, called “Delay and Sum Beamformer“ accomplishes the spatial filtering task via - as the name suggests - summing the sensor outputs after a particular delay [15]. The first step of Delay-and-Sum beamforming is time-shifting each sensor signal by a value corresponding to the TDOA between that sensor and the reference one

$$\begin{aligned} y_{a,i} &= y_i\,(t+\tau_i) \\ &= \alpha_i s(t-T_0) + e_{a,i}(t) \\ &= x_{a,i}(t) + e_{a,i}(t),\ \ i = 1, 2, ..., L \end{aligned} \tag{2.28}$$

where $e_{a,i} = e_i\,(t+\tau_i)$. The next step is just adding up these time-shifted sensor outputs, which results in

$$\begin{aligned} z(t) &= \frac{1}{L}\sum_{i=1}^{L} y_{a,i}(t) \\ &= \alpha_s s(t-T_0) + \frac{1}{L} e_s(t) \\ where\ \alpha_s &= \frac{1}{L}\sum_{i=1}^{L}\alpha_i \\ e_s(t) &= \sum_{i=1}^{L} e_{a,i}(t) \\ &= \sum_{i=1}^{L} e_i\,(t+\tau_i) \end{aligned} \tag{2.29}$$

The input SNR, based on the model given in (2.27) of the receiving array is (relative to the i^{th} signal),

$$SNR_i = \frac{\sigma_{x_i}^2}{\sigma_{e_i}^2} = \alpha_i^2 \frac{\sigma_s^2}{\sigma_{e_i}^2} \tag{2.30}$$

where $\sigma_{x_i}^2 = E[x_i^2]$, $\sigma_{e_i}^2 = E[e_i^2]$, and $\sigma_s^2 = E[s^2(t)]$ are the variance of the signals x_i(t), e_i(k) and s(t) respectively.

After Delay-and-Sum processing, the SNR can be expressed as the ratio of the variances of the first and second terms in the right hand side of (2.29):

$$
\begin{aligned}
SNR_{out} &= L^2\alpha_s^2 \frac{E\left[s^2(t-T_0)\right]}{E[e_s^2]} = L^2\alpha_s^2\frac{\sigma_s^2}{\sigma_{e_s}^2} \\
&= \left(\sum_{i=1}^{L}\alpha_i\right)^2 \frac{\sigma_s^2}{\sigma_{e_s}^2}, \\
where\ \sigma_{e_s}^2 &= E\left\{\left[\sum_{i=1}^{L} e_i\left[t+\tau_i\right]\right]^2\right\} \\
&= \sum_{i=1}^{L}\sigma_{e_i}^2 + 2\sum_{i=1}^{L-1}\sum_{j=i+1}^{L}\varsigma_{e_ie_j},
\end{aligned}
\tag{2.31}
$$

with $\sigma_{e_i}^2 = E[e_i^2]$ is the variance of the noise signal e$_i$(t) and $\varsigma_{e_ie_j} = E\left\{e_i\left[t+\tau_i\right]e_j\left[t+\tau_i\right]\right\}$ is the cross-correlation between e$_i$(t) and e$_j$(t)

The beamformer is regarded as successful if the condition

$$SNR_{out} > SNR_i \tag{2.32}$$

is satisfied

1st particular case : Assume that noise signals at the elements of receiving array are uncorrelated, i.e. $\varsigma_{e_ie_j} = 0, \forall i, j = 1, 2, ..., L,\ i \neq j$ and they all have the same variance, i.e. $\sigma_{e_{i_1}}^2 = \sigma_{e_{i_2}}^2 = ... = \sigma_{e_{i_L}}^2$. And also assume that all the attenuation factors are equal to 1, i.e. $\alpha_i = 1, \forall i$. Then

$$SNR_{out} = N \times SNR_i \tag{2.33}$$

So, it is interesting that under the particular condition, simply delaying and summing the sensor outputs results in an SNR improvement by a factor equal to the number of sensors.

2nd Particular Case : Here, an assumption which is more restrictive than the 1st one will be made. Assume that the noise signals have the same energy (they may be correlated), and all the attenuation factors are 1 as in the 1st case. In this case;

$$SNR_{out} = \frac{L}{1+\rho_s} \times SNR_i$$
$$where\ \rho_s = \frac{2}{L}\sum_{i=1}^{L-1}\sum_{j=i+1}^{L}\rho_{e_ie_j}, \tag{2.34}$$
$$\rho_{e_ie_j} = \frac{\varsigma_{e_ie_j}}{\sigma_{e_i}\sigma_{e_j}}$$

$\rho_{e_ie_j}$ is the correlation coefficient with $\left|\rho_{e_ie_j}\right| \leq 1$. If the noise signals are completely correlated, i.e. if the correlation coefficient $\rho_{e_ie_j} = 1$, then $SNR_{out} = SNR_i$, so no gain is possible with delay-and-sum technique in this condition. Until this point, the performance of the beamformer is investigated in terms of SNR improvement. Another way of illustrating the performance is through examining the beam pattern (or directivity pattern or spatial pattern). Note that the Delay-and-Sum Beamformer is actually an L-point spatial filter and its beam pattern is defined as the magnitude of the spatial filter‘s directional response. Now, consider the Uniform Linear Array in Figure 2.6, TDOA between the ith and the reference sensors is

$$\tau_i = \frac{(i-1)dcos(\phi)}{c} \tag{2.35}$$

For the Uniform Linear Array case, the directional response of the DS filter can be expressed as

$$S_{DS}(\psi,\phi) = \frac{1}{L}\sum_{i=1}^{L}\left[e^{\frac{j2\pi(i-1)fdcos\theta}{c}}\right]e^{-\frac{j2\pi(i-1)fdcos\psi}{c}}$$
$$= \frac{1}{L}\sum_{i=1}^{L}e^{-\frac{j2\pi(i-1)fd[cos\psi - cos\phi]}{c}} \tag{2.36}$$

where ψ $(0 \leq \psi \leq \pi)$ is a directional angle. The beam pattern is then written as

$$A_{DS}(\psi,\phi) = |S_{DS}(\psi,\phi)|$$
$$= \left|\frac{sin\left[L\pi fd\left(cos\psi - cos\phi\right)/c\right]}{Lsin\left[\pi fd\left(cos\psi - cos\phi\right)/c\right]}\right| \tag{2.37}$$

For the implemented acoustic radar system, the beamformer response is given in Figure 2.7, with the pulse frequency f is chosen as 1417 Hz and the inter-element spacing of both trans-

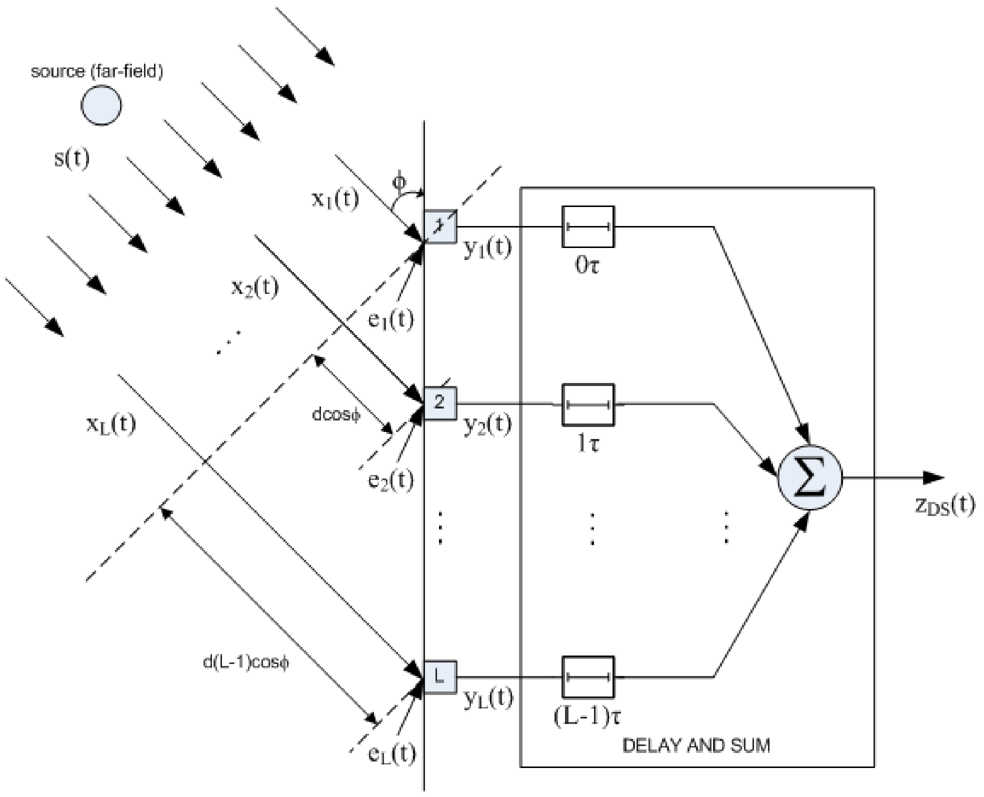


Figure 2.6: ULA Delay-and-Sum Beamforming

mit and receive array is chosen as 12 cm (the reason why these values are chosen will be explained).

Consider Figure 2.7. The beam which has the highest amplitude is called mainlobe, all the others are sidelobes. An important parameter for a beamformer design is defined as beamwidth, which is the region between first zero-crosses on either side of the mainlobe. Beamwidth can easily be calculated considering (2.37). Equating the numerator to zero (with the steer angle ϕ is chosen as 90°), the angle at which beamformer response becomes zero can be easily calculated such that;

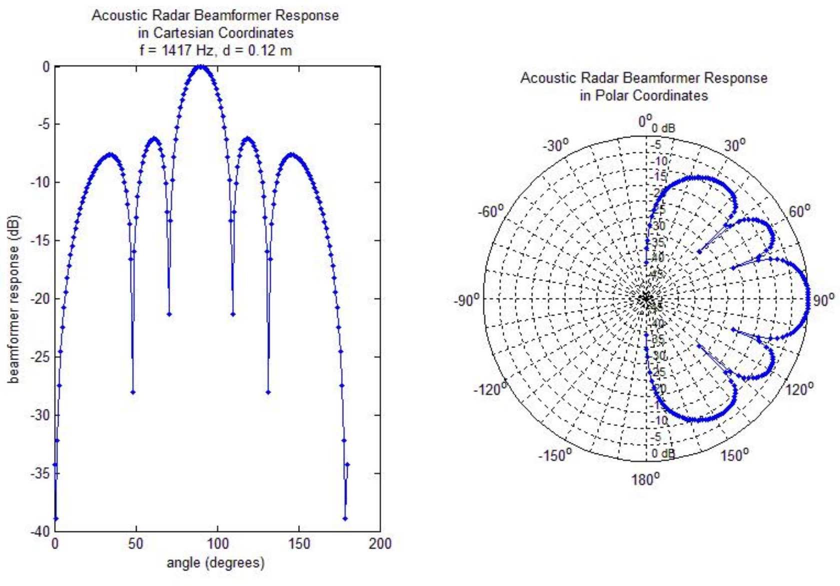


Figure 2.7: Directivity response of the Acoustic Radar Beamformer with steer angle chosen as 90 ° (Same for transmit and receive arrays)

$$\begin{aligned}
sin\left[L\pi fd\left(cos\psi - cos\phi\right)/c\right] = 0 \rightarrow \\
\frac{L\pi fdcos\psi}{c} = \pi \rightarrow \\
\psi = cos^{-1}\frac{c}{Lfd} \\
note\ that\ beamwidth\ = 2\times\psi \rightarrow \\
beamwidth = 2cos^{-1}\frac{c}{Lfd}
\end{aligned} \tag{2.38}$$

Obviously, the performance of a beamformer with a narrower beamwidth is considered as good. Beamwidth decreases as the inter-element spacing d increases and the frequency of the transmitted/received signal is increased. It should be noted that there is a limit on the inter-element spacing. One would like to increase d as large as possible to get a narrower beamwidth and more noise reduction. But, when d is larger than $\lambda/2$ = c/2f, where λ is the wavelength of the signal, spatial aliasing would arise. So, there is a trade-off between d, f and the beamwidth of the beamformer. In order to avoid spatial aliasing, the condition $d \leq \lambda/2 = c/2f$ must be satisfied. In the particular implementation, the maximum possible frequency satisfying the Spatial Sampling Theorem (i.e. not causing spatial aliasing) is 1417 Hz. It is achieved by minimizing the inter-element spacing d, which is, by physical constraints of the loudspeaker array elements, chosen as 12 cm.

2.2.2 Designing the Maximum SNR Spatial Filter [15]

In this section, the optimal spatial filter which maximizes the SNR at the output of the beamformer is [15] presented.

Equation (2.28) may be rewritten in vector form as;

$$\begin{aligned} \mathbf{y}_a(k) &= s(t-T_0)\boldsymbol{\alpha} + \mathbf{e}_a(t) \\ where\ \mathbf{y}_a(t) &= \left[y_{a,1}(t)\ y_{a,2}(t)\ ...\ y_{a,L}(t)\right]^T, \\ \mathbf{e}_a(t) &= \left[e_{a,1}(t)\ e_{a,2}(t)\ ...\ e_{a,L}(t)\right]^T, \\ \boldsymbol{\alpha} &= [\alpha_1\ \alpha_2\ ...\ \alpha_L]^T \end{aligned} \tag{2.39}$$

where the subscript “a“ is used to indicate that the corresponding signal is “delayed“ (in (2.28), this was also indicated by “a“ subscript). Under the assumption that the signal and noise are uncorrelated, the correlation matrix of $\mathbf{y}_a$(t) can be expressed as

$$\mathbf{R}_{y_a y_a} = \sigma_s^2 \boldsymbol{\alpha}\boldsymbol{\alpha}^T + \mathbf{R}_{e_a e_a} \tag{2.40}$$

where $\mathbf{R}_{e_a e_a} = E\left[\mathbf{e}_a(t)\mathbf{e}_a^T(t)\right]$ is the noise correlation matrix. Note that in a simple Delay-and-Sum beamformer, the time-shifted sensor outputs are simply summed. But, in a more general beamformer, the time-shifted outputs are weighted before summation, this may be expressed as;

$$\begin{aligned} z(t) &= \mathbf{h}^T \mathbf{y}_a(t) \\ &= \sum_{i=1}^{L} h_i y_{a,i}(t) \\ &= s(t-T_0)\mathbf{h}^T \boldsymbol{\alpha} + \mathbf{h}^T \mathbf{e}_a(t) \\ where\ \mathbf{h} &= [h_1\ h_2\ ...\ h_L]^T \end{aligned} \tag{2.41}$$

Delay-and-Sum beamformer is a special case of this filter where h$_n$‘s are taken as 1/N, i.e. no weighting is employed, time-delayed signals are simply summed in Delay-and-Sum beamformer). Using (2.41), the output SNR may be written as

$$SNR(\mathbf{h}) = \frac{\sigma_s^2 \left(\mathbf{h}^T \boldsymbol{\alpha}\right)^2}{\mathbf{h}^T \mathbf{R}_{e_a e_a} \mathbf{h}} \tag{2.42}$$

Maximizing the SNR at the output of the beamformer is equivalent to solving the following eigenvalue problem:

$$\sigma_s^2 \boldsymbol{\alpha}\boldsymbol{\alpha}^T \mathbf{h} = \lambda \mathbf{R}_{e_a e_a} \mathbf{h} \tag{2.43}$$

Assuming that $\mathbf{R}_{e_a e_a}^{-1}$ exist, the optimal solution to the above eigenvalue problem is the eigenvector, $\mathbf{h}_{max}$ corresponding to the maximum eigenvalue, λ_{max} of $\sigma_s^2 \mathbf{R}_{e_a e_a}^{-1} \boldsymbol{\alpha}\boldsymbol{\alpha}^T$ so;

$$\begin{aligned} z_{max}(t) &= \mathbf{h}_{max}^T \mathbf{y}_a(t), \\ SNR(\mathbf{h}_{max}) &= \lambda_{max} \end{aligned} \tag{2.44}$$

Now, remember the *1st particular case* examined in 2.2.1, in which noise signals are uncorrelated, they all have the same variance and the attenuation factors (σ_i's) are all 1. Then, for this particular case, (2.44) becomes

$$SNR.\boldsymbol{\alpha}\boldsymbol{\alpha}^T \mathbf{h}_{max} = \lambda_{max} \mathbf{h}_{max} \tag{2.45}$$

Left multiplying both sides by $\boldsymbol{\alpha}^T$,

$$\begin{aligned} \lambda_{max} &= N.SNR_i \ so \ that, \\ SNR_i(\boldsymbol{h}_{max}) &= N.SNR_i = SNR_{out} \end{aligned} \tag{2.46}$$

This implies that

$$\mathbf{h}_{max} = \frac{1}{N} [1 \ 1 \ ... \ 1]^T \tag{2.47}$$

So, for the 1th particular case given in section 2.2.1,which is also the assumption in this Acoustic Radar Implementation, the maximum SNR filter is identical to the Delay-and-Sum beamformer.

CHAPTER 3

ACOUSTIC RADAR

In this chapter, a general description of the acoustic radar system is given. The detailed technical description of the implemented Acoustic Radar System is given in CHAPTER 4.

3.1 System Structure

The basic structure of the implemented acoustic radar system is given below in Figure 3.1. The diagram shows a conceptual structure, rather than a functional block diagram of the FPGA implementation (the functional block diagram of the implementation is given in the following chapter).

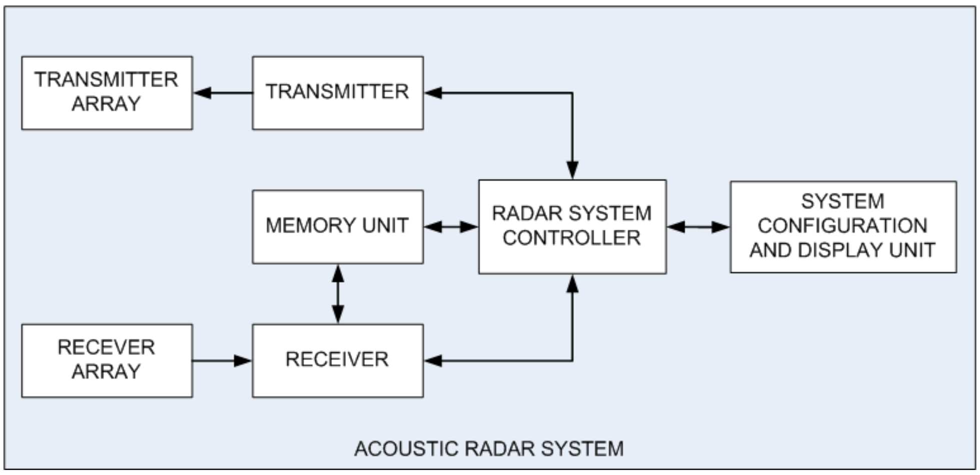


Figure 3.1: Acoustic Radar System Structure

3.1.1 Radar System Controller

Radar system controller is the main processor of the system which is responsible for operating the whole system synchronously and organizing the data flow throughout the system. Its operation is managed by a finite state machine which provides the control of the system for standalone operation.

The functionality of the controller may be described briefly as follows: First, the controller gives instruction to the transmitter unit for generation of a pulse which is to be sent to air ultimately from the transmit array. To generate pulses for the transmit array elements, the only information that the transmitter unit needs is the steer angle. So, the only action that radar system controller takes to send a pulse to air is just triggering the transmitter unit with the desired steer angle information. Acoustic Radar System has a scan ability in range [0, 180] degrees with a resolution of 1 degrees. However, in practice, a coarse scan step is selected since the array beamwidth is large. After triggering the transmit pulse generation with a particular steer angle request, acoustic radar system controller runs the receive unit which consists of a delay-sum beamformer and a correlator unit. Upon triggering receiver unit, system controller waits until correlator finishes its operation. Correlator provides the radar information to the system controller, which are namely;

- the **index** of the correlation maximum, which is directly related to the fly time of the radar pulse in the air

- the **value** of the correlation maximum, which is directly related to reflectivity quality of the scanned object

Upon gathering this information from correlator, acoustic radar system controller sends it to Display Unit (together with current steer angle information attached) which is ultimately to be displayed on the Radar Screen. After the position information of a particular look direction is plotted on the radar screen, the Display Unit sends a feedback signal to System Controller with which it is triggered to scan the next steer angle.

3.1.2 Transmitter

Transmitter block is mainly responsible for generating the transmit pulse. This block is a slave unit which is controlled by the main controller of the system, namely the “Radar System Controller“.

The main controller of the system does not deal with the details of the pulse generation process, i.e., all the necessary steps for transmit beam steering is handled by the Transmitter block. This unit implements transmit beam steering digitally [14]. Transmit beam steering is similar to the receive beamforming which is explained in 2.2.1. The difference is that in delay-sum receive beamforming, the received waves are summed together after passing them through proper delays, however, in transmit beam steering, the reverse process applies, namely, the waves are transmitted to the air after passing them through very same delays as in the receive case. How transmit beam steering is achieved is illustrated in Figure 3.2. Note from the figure that feeding the transmit array elements with signals having phase differences causes the transmitted wavefront to be directed to a particular direction.

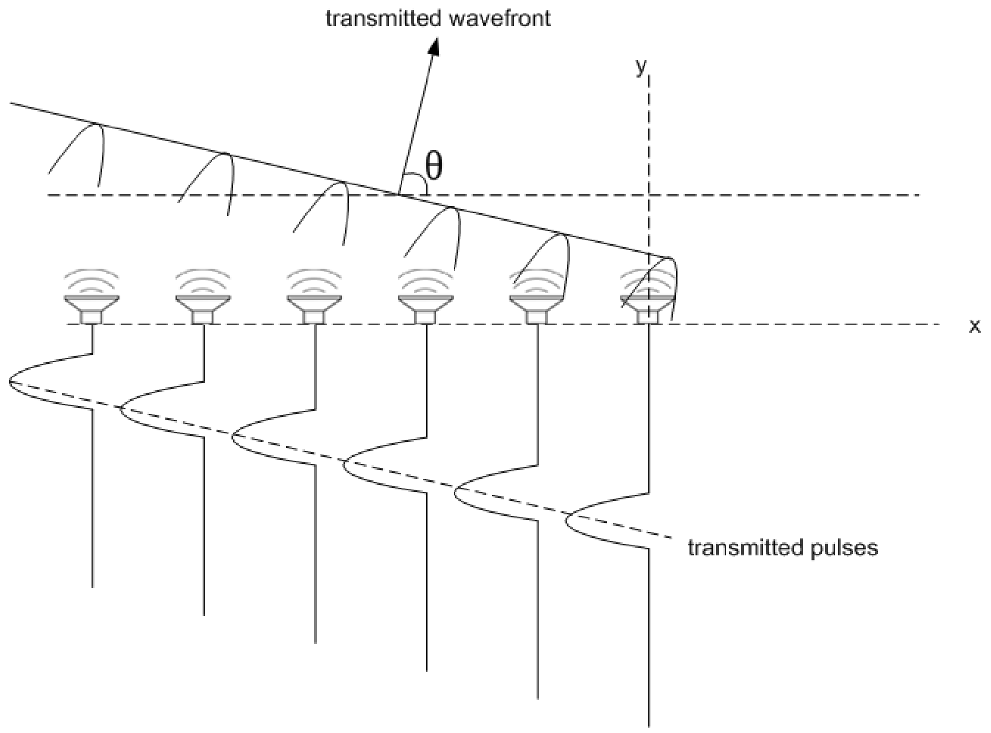


Figure 3.2: Phased Array Transmit Steer

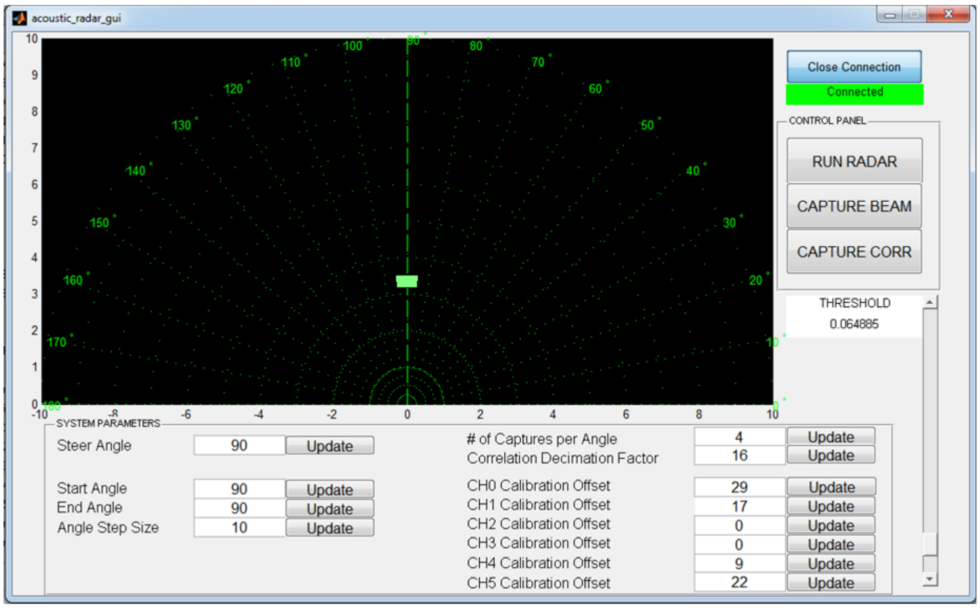
3.1.3 Receiver

This block is responsible to receive the expected acoustic echo pulses from the air (which are captured by the receive array elements, i.e. the microphones) and then to apply matched-filtering after receive delay-and-sum beamforming. This block, just like the Transmitter, is controlled by the main system controller, i.e. Radar System Controller.

The receiver block mainly consists of three parts, data capture and filtering part, delay-and-sum beamforming part and correlator part. Data capture and filtering part operates independently from the System Controller, it continuously captures data from 6-Channel ADC - with a sampling rate of 250 MSPS for each channel-, removes the DC present in the signal, low-pass filters it and ultimately dumps the filtered data to 6 input channel FIFOs for further processing. The other two fundamental units of the receiver, namely delay-and-sum beamformer and correlator units, are run by the System Controller. The process is as follows: at first, the 6 channel FIFOs mentioned above are held on reset by the System Controller, so write accesses of the data capture and filtering part to these FIFOs are simply ignored. Then, the reset of the FIFOs is released, at the same time delay-and-sum beamformer part is triggered with the current steer angle information. Delay-and-sum beamformer starts its operation, it sums 6 channel data with the proper delays applied to each channel and writes the resulting beam data to the Memory Unit. The capture length is so long that it is enough to span the entire range of the radar. Upon reliazing that the operation of the delay-and-sum beamformer is completed, System Controller triggers correlator part of the receiver. Correlator starts to process the data which is already recorded to the Memory Unit by the delay-and-sum beamformer. After correlation is over, it detects the index and magnitude of the peak, which are supplied to System Controller to be ultimately sent to Display Unit.

3.1.4 System Configuration and Display Unit

This block, which is actually a MATLAB GUI, forms the visual part of the system. In addition to this, it provides user the ability to adjust the acoustic radar system parameters. System Configuration and Display Unit is connected to the Acoustic Radar System Controller over a UDP (User Datagram Protocol) link. Upon a click to an update button in the GUI to modify a system parameter, a UDP packet is transmitted to the Acoustic Radar System Controller, which is realized as a register write access ultimately. In the normal radar operation, it receives a UDP packet for the current look direction. This packet consists of steer angle information, the index and the value of the correlation. After a simple threshold comparison, the decision whether a valid object is detected for the current angle is made and the corresponding area on the plot is highlighted with a proper intensity which is proportional to the magnitude of the correlation. Upon displaying the position information of the current steer angle, System Configuration and Display Unit transmits a feedback packet to the System Controller, which triggers it to start another transmit-receive cycle for the next steer angle.



THRESHOLD	0.064885
Steer Angle	90
Start Angle	90
End Angle	90
Angle Step Size	10
# of Captures per Angle	4
Correlation Decimation Factor	16
CH0 Calibration Offset	29
CH1 Calibration Offset	17
CH2 Calibration Offset	0
CH3 Calibration Offset	0
CH4 Calibration Offset	9
CH5 Calibration Offset	22

Figure 3.3: Acoustic Radar Display Unit

3.1.5 Memory Unit

This unit is a block memory used to dump the received echo for further processing. It is implemented using FPGA's dedicated memory primitives (detailed explanation is given in Section 4). This memory is used by three main units in the acoustic radar system, which are, namely, *delay-and-sum beamformer*, *correlator* and the *system controller* units. This multi-purpose usage is achieved via a bus arbitration structure such that for a specific time, the physical port of the memory is devoted to one of the candidates (the bus owner is assigned by the *system controller*). *Delay-and-sum beamformer* and *correlator* uses this memory both as an input source and an output sink. They read the memory for the input data and write their output to it. So, this is an obvious memory saving. During radar operation, *system controller* does not access to the memory. But, when beamformer or correlator output is asked to be captured by *System Configuration and Display Unit*, *system controller* accesses the memory and sends the requested data to pc.

3.1.6 Transmit and Receive Arrays

Transmit Array is responsible to convert electrical signals which are supplied by the Transmitter unit to acoustical signals. It consists of 6 loudspeakers. The operation of the Receive Array is just the opposite of the Transmit one, which is converting acoustic waves to electrical signals, which are then fed to the Receiver block for further processing. It consists of 6 microphones.

3.1.7 Acoustic Radar System Parameters

In a radar system design problem, there are several parameters which should be taken into account carefully. Radar performance is determined by the design parameters.

In this thesis, only a subset of these parameters is considered since the implemented system has the ability to scan the range and azimuth only. The radar design parameters used in designing the Acoustic Radar System can be listed as follows,

- Pulse Width
- Wavelength
- Pulse Repetition Period
- Pulse Shape

These design parameters -implicitly or explicitly- appears in the so-called “radar equation“ and they affect the maximum detection range of the system. So, before explaining them, it would be convenient to give an insight on “radar equation“.

3.1.7.1 The Radar Equation

The radar equation is basicly the measure on the expected return signal and SNR [17]. Assume that a radar transmits a pulse with power P_T and there is a target at a range R. Assume that the radar transmitting antenna has an isotropic radiation pattern, then the power would spread with spherical symmetry. Then the power flux per unit area at range R would be

$$Power\ Density\ = \frac{P_T}{4\pi R^2} \tag{3.1}$$

If the antenna has a gain G then the power density at the target area would be multiplied by G. Assume that the effective area of the target is α, then the reflected power from the target would be

$$Reflected\ Power = \frac{P_T G \alpha}{4\pi R^2} \tag{3.2}$$

and since the power is reflected isotropically, the reflected power density back at the radar is

$$Reflected\ Power\ Density = \frac{P_T G \alpha}{(4\pi R^2)^2} \tag{3.3}$$

Let A be the effective area of the radar's receiving antenna, the the power received by the antanna is given by

$$P_R = \frac{P_T G A \alpha}{(4\pi R^2)^2} \tag{3.4}$$

The relationship between the antenna gain G and its effective area A is given by

$$A = \frac{G\lambda^2}{4\pi} \tag{3.5}$$

Insert (3.5) in (3.4), the basic radar equation is obtained

$$P_R = \frac{P_T G^2 \lambda^2 \sigma}{(4\pi)^3 R^4} \tag{3.6}$$

Usually, the signal that is returned when the target lies somewhere along the maximum of the radar beam is of particular interest. Considering this target, let's define the maximum gain of the antenna G_o as [16]:

$$G_o = \frac{4\pi A f}{\lambda^2} \tag{3.7}$$

Insert (3.7) in (3.6), then

$$P_R = \frac{P_T \sigma A^2 f^2}{4\pi R^4 \lambda^2} \tag{3.8}$$

Suppose the minimum power required for satisfactory detection, $P_{R_{min}}$ is known, so (3.8) may be solved for the maximum range of detection, R_{max};

$$R_{max} = \sqrt[4]{\frac{P \sigma A^2 f^2}{4\pi P_{R_{min}} \lambda^2}} \tag{3.9}$$

(3.9) is function of the system design parameters which are to be explained shortly.

3.1.7.2 Pulse Width

This parameter has a direct effect on the minimum detectable received signal power of the radar, which appears in (3.9) as $P_{R_{min}}$. $P_{R_{min}}$ varies inversely with pulse length since the receiver bandwidth is an inverse function of the pulse length. There is also a tradeoff between increasing the transmitted power through increasing the pulse width and decreasing range ambiguity of the radar. If Pulse Width is increased, the transmitted power also increases. However, increasing the pulse width also increases the range ambiguity (in other words, decreases the range resolution). The range resolution of a radar system is defined as

$$Range\ Resolution = \frac{c\tau}{2} \tag{3.10}$$

where, τ is the pulse length. In our particular implementation, pulse width is chosen as 2 milliseconds, resulting in a range resolution of $\frac{(340\ m/s)\times(2\ msec)}{2}$ = 34 cm.

3.1.7.3 Wavelength

Wavelength has a direct effect on the power of received signal (3.8). Attenuation increases as the frequency of the signal is increased, in other words, the wavelength is decreased. Beamwidth is another important parameter which is affected by the wavelength (2.38). Beamwidth decreases with increasing frequency. There is a maximum limit on the frequency of the pulse, which is spatial sampling rate constraint. In order to avoid spatial aliasing, the array spacing has to satisfy

$$d \leq \frac{\lambda}{2} = \frac{c}{2f} \tag{3.11}$$

according to the spatial sampling theorem.

So, from (3.11), it is obvious that there is a limit on the maximum frequency of the wave,

$$f_{max} = \frac{c}{2d} \tag{3.12}$$

There are also physical constraints on the distance between transmitter array elements. The minimum distance between transmitter array elements, i.e., between the loudspeakers is 0.12 m. So, frequency of the radar pulse is chosen as the maximum possible $f = \frac{c}{2d} = \frac{340}{2*0.12} =$ 1417 Hz

3.1.7.4 Pulse Repetition Period

Pulse Repetition Period (or Frequency) is the time between radar pulses. PRF determines the Maximum Unambiguous Range (3.13) of the system.

$$Maximum\ Unambiguous\ Range = \frac{c \times PRF}{2} \tag{3.13}$$

The Maximum Unambiguous Range is the longest range to which a transmitted pulse can travel and return to the radar before the next pulse is transmitted. “Before the next pulse is transmitted“ is the keyword in the definition, because this implies that it is the maximum distance that the radar system measures without any ambiguity. In our particular application the Maximum Unambiguous Range is chosen as 10 meters. As a result, the Pulse Repetition Period of the system $T = \frac{2\times(10\ m)}{(340\ m/sec)} = 58.8\ msec$

3.1.7.5 Pulse Shape

Pulse shape selection is an important decision in the radar system design process. In practice, generally used radar waveforms are linear chirps (chirp is a signal whose frequency increases or decreases with time), non-linear chirps, stepped frequency waveforms (SFW) and binary phase coded pulses (like gold sequences and barker codes). Chirp signals are mostly used to improve the range resolution of the system via pulse compression. Phase coded pulses are preferred for thesis desirable autocorrelation functions. In this thesis work, a simple pulse shape is used. The reason why such a simple pulse shape is selected is to have simple transmitter and receiver structures (Note that the primary task of this thesis work is not to implement sophisticated array signal processing algorithms). Acoustic radar pulse is generated by windowing a sinusoidal signal with a gaussian, as seen from Figure 3.4.

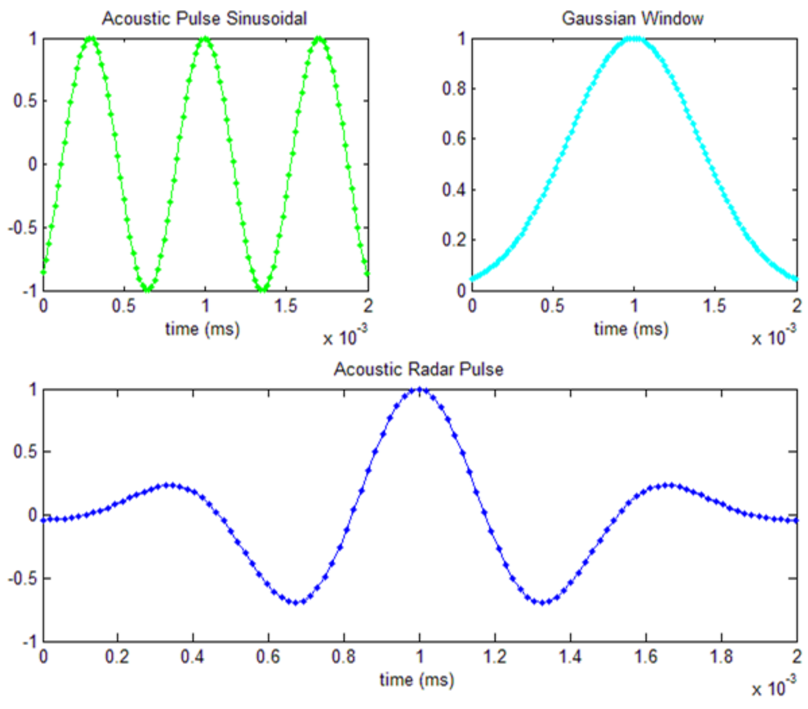


Figure 3.4: The Acoustic Radar Pulse

The magnitude response (in dB) of the acoustic radar pulse, which is the spectrum of a sinusoidal signal windowed by a gaussian, is given in Figure 3.5. It can be seen from the figure that the carrier frequency of the acoustic radar is 1417 Hz (approximately) and the 3 db bandwidth of the signal is 671 Hz. The carrier frequency and the bandwidth of the pulse is comparable. So, narrowband assumption, which requires B $\ll \omega_c$ condition, cannot be applied for this case. In other words, the acoustic radar pulse is a wideband signal.

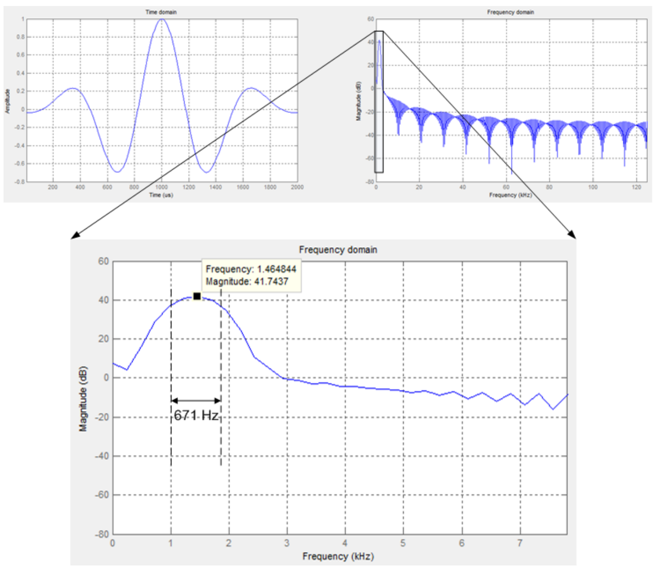


Figure 3.5: Acoustic Radar Pulse Magnitude Response

Before leaving this section, the **Ambiguity Function**, which is a well-known performance parameter for a radar system, corresponding to the particular Acoustic Radar Pulse will be given. The ambiguity function (AF) represents the time response of a filter which is matched to a given finite energy signal when the signal is received with a delay τ and a Doppler shift ν relative to the nominal values (zeros) expected by the filter [18]. It is defined as;

$$|\chi(\tau, \nu)| = \left| \int_{-\infty}^{\infty} u(t)u^*(t+\tau)e^{j2\pi\nu t}dt \right| \tag{3.14}$$

where u is the complex envelope of the signal. Using (3.14), AF for the Acoustic Radar Pulse is calculated. It is shown in Figure 3.6. Note that the *Doppler* and *Delay* variables in Figure 3.6 are normalized values with respect to the pulse duration (or width), which is 2 msec as mentioned above.

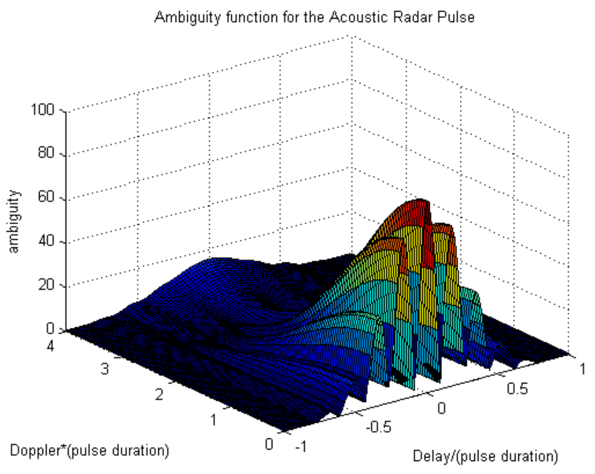


Figure 3.6: Ambiguity function for the Acoustic Radar Pulse

The ambiguity performance of the Acoustic Radar Pulse can be evaluated better when compared with different pulse shapes. In Figure 3.7, ambiguity functions for two different signals are given. The signal whose AF is given on the left is an unmodulated (rectangular) pulse. The other one is a linear chirp pulse, whose frequency varies linearly from 100 Hz to 1417 Hz (which is the frequency of the Acoustic Radar Pulse). They both have the same pulse width of 2 msec as the Acoustic Radar Pulse. When the mainlobe widths of three pulses are compared, it is easily concluded that the linear chirp pulse has the best performance, while the unmodulated pulse has the worst. The ambiguity performance of the particular Acoustic Radar Pulse is in between, as seen easily from figures 3.6 and 3.7.

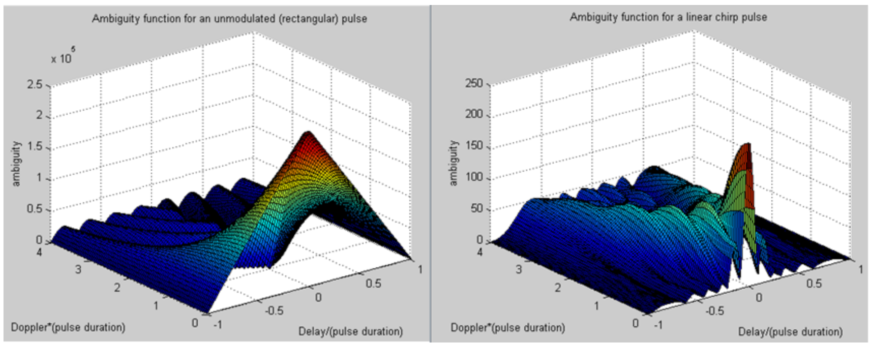


Figure 3.7: Ambiguity function for an unmodulated (left) pulse and a chirp pulse (rigth) of the same length as the Acoustic Radar Pulse

CHAPTER 4

FPGA IMPLEMENTATION OF RADAR PROCESSING

In this section, the detailed description of the implemented radar system is given. Firstly, the hardware on which the system is implemented will be explained briefly. Then the FPGA building blocks will be described in detail. Note that all the mathematical functions of the Acoustic Radar is implemented in real time on FPGA.

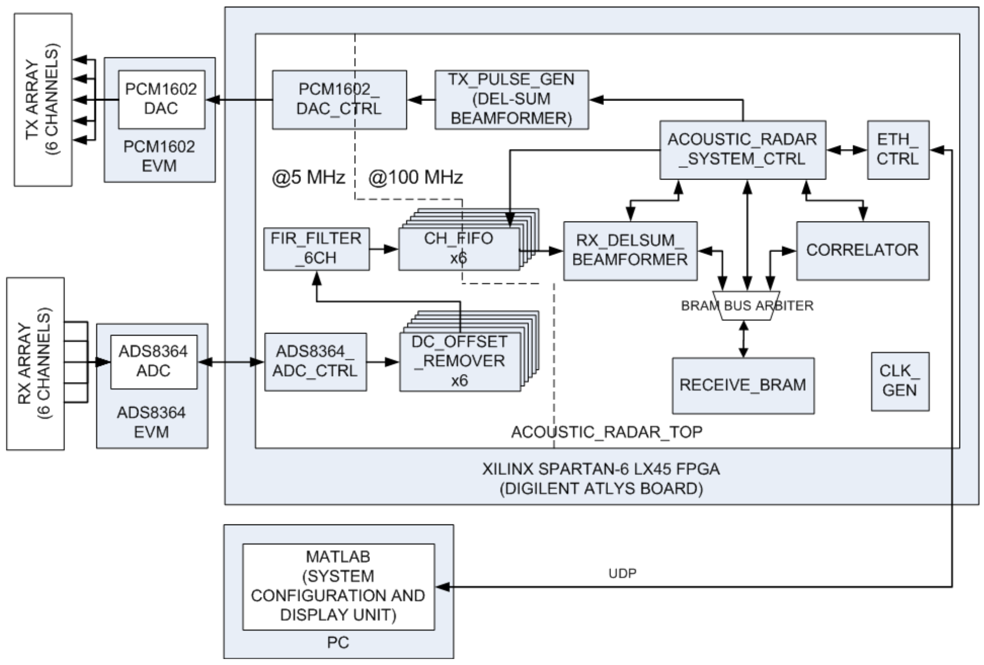


Figure 4.1: Implemented Radar System Structure

4.1 Hardware Description

In Figure 4.1, the radar system structure is given. The hardware that the radar system is implemented on mainly consists of 3 parts:

1. DIGILENT ATLYS Spartan-6 FPGA Board [20]
2. TI PCM1602 DAC Evaluation Module [21]
3. TI ADS8364 ADC Evaluation Module [22]

4.1.1 DIGILENT ATLYS Spartan-6 FPGA Board

The main hardware unit of the radar system is the FPGA board, namely, Digilent ATLYS XILINX Spartan-6 Board. Figure 4.2 shows this board and the functional outline of the FPGA. All the digital signal processing is handled in the FPGA. FPGA Board has external interfaces with ADC and DAC evaluation modules and the PC. The reason why this FPGA board is chosen is that the performance of Spartan-6 FPGA is enough for our particular application and that the board is cost efficient when compared to other available ones in the field.

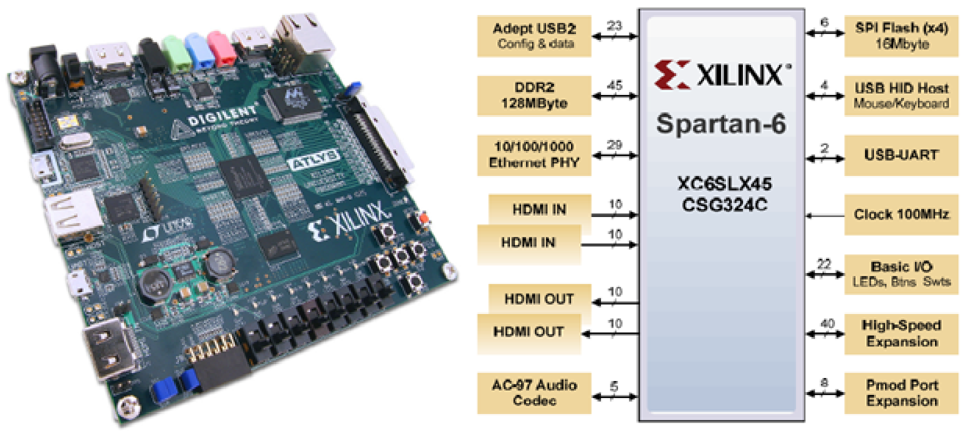


Figure 4.2: Top view and functional block diagram of ATLYS FPGA Board

The Spartan-6 LX45 is a high performance FPGA [19], it consists of

- 6822 slices (each of them contains four 6-input LUTs and eight flip flops)

- 2.1 Mbits of block RAMs (which is really important as the system necessitates many memory elements like FIFOs, RAMs and ROMs)
- Four clock tiles (8 DCMs (Digital Clock Manager) and 4 PLLs (Phased Locked Loops))
- 58 DSP Slices (these are specific optimized structures in FPGA, supporting many DSP algorithms with minimal use of the general purpose FPGA fabric. They are, at first look, 18x18 bit two‘s complement multiplier followed by a 48 bit sign extended adder/substractor/accumulator, which are configurable in various ways)
- 500 MHz+ clock speeds (operating FPGA in such high speeds is not a good practice, not realistic actually. For our system 100 MHz is a far enough operating frequency as acoustical waves are extremely slow compared with radio waves)

The above properties of the FPGA board as well as its price makes it a reasonable choice for our particular application.

As shown in Figure 4.1, ATLYS board has interfaces with other hardware units in the system. It is connected to the ADC Evaluation Module via the 8 pin PMOD port expansion header and to the DAC Evaluation Module via the 40-pin high speed expansion header. The connection to the PC is over an Ethernet cable (the communication between the PC and FPGA is implemented using UDP protocol). DAC EVM connection is composed of 6 copper wires. But ADC EVM connection is not that simple, because the 40-pin expansion header port of the FPGA board and the expansion connectors of the ADC EVM are quite incompatible with each other. To connect these two hardware units, a custom connection PCB is designed and produced (it is shown in Figure 4.3).

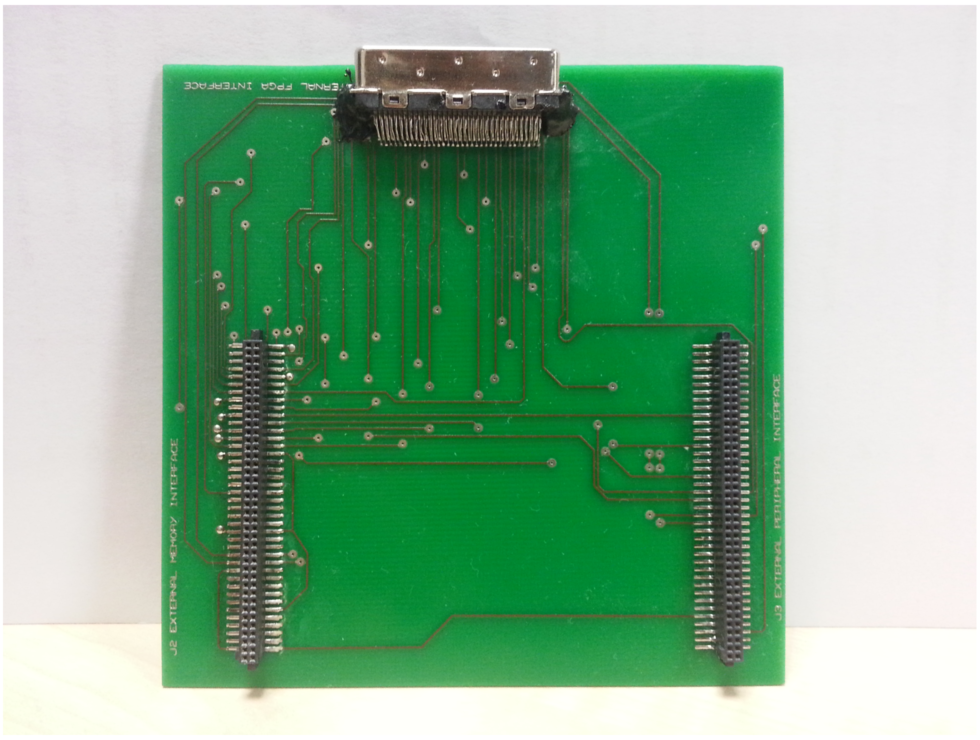

J2 EXTERNAL MEMORY INTERFACE
J3 EXTERNAL PERIPHERAL INTERFACE

Figure 4.3: PCB for connecting ATLYS Board to ADS8364 ADC EVM

4.1.2 TI PCM1602 DAC Evaluation Module

Acoustic radar system employs a 6 channel audio DAC from Texas Instruments (PCM1602, 24-Bit, 192-kHz Sampling, 6 Channel, Enhanced Multilevel, Delta-Sigma Digital-to-Analog Converter) to drive the transmitter array elements (which is composed of 6 loudspeakers). The interface between FPGA and DAC is a serial audio data interface (the detailed description is given in Section 4.2.2). PCM1602 DAC Evaluation Module whose functional block diagram is given in Figure 4.4 is a demonstration board which provides an ideal platform for evaluating the performance and operation of the PCM1602 audio Digital-to-Analog Converter. This is mainly composed of 2 parts,

- The carrier card: A digital audio receiver is used to have an interface with an audio source (electrical or optical) generating a stereo, linear PCM, S/PDIF data stream. The audio receiver provides the user the opportunity to connect a commercial audio product and evaluate the performance of the DAC; it isolates the user from handling the serial audio interface signalling. There are three second order Butterworth low-pass filters

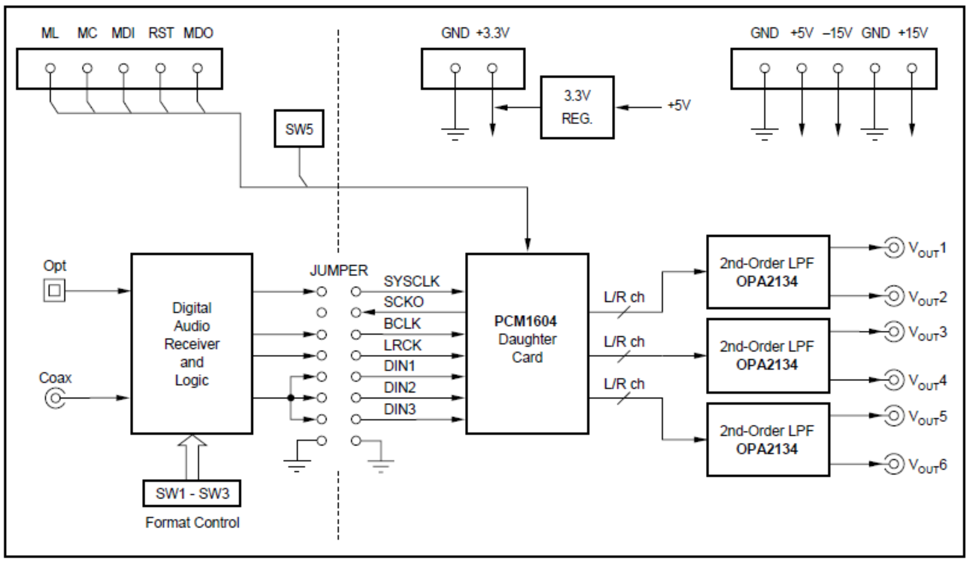


Figure 4.4: PCM1602 DAC Evaluation Module Functional Block Diagram

for the DAC outputs and audio output connectors to interface with commercial loudspeakers. Carrier card has also a printer port which provides the user to access (write and read) the PCM1602 registers through the demonstration software running on the PC (this printer port interface is used to access the user programmable mode control registers).

- PCM1602 Daughter Card: It is a small printed circuit board which is connected to the carrier card via four male connectors and on which PCM1602 audio DAC is mounted.

The main hardware component of the acoustic radar system, namely, ATLYS FPGA board uses the digital audio interface of this PCM1602 EVM. As explained above, by default, the DAC EVM is shipped with an on board digital audio receiver and that receiver drives the audio DAC. For our particular application, the digital audio interface is separated from the on-board digital audio receiver by just removing the jumpers on the EVM carrier card (note the jumper in Figure 4.4) and connected to the ATLYS FPGA board via 6 copper wires.

4.1.3 TI ADS8364 ADC Evaluation Module

ADS836 EVM, whose top view is given in Figure 4.5, is a full feature evaluation board for the ADS8364 250 KHz, 16 Bit, 6-channel simultaneous sampling analog-to-digital converter. It is possible to connect this EVM to C5000 and C6000 DSK platforms directly through two 80-pin interface connectors. Being designed for easy interfacing with TI‘s C5000 and C6000 DSP platforms, it is not that straightforward to connect this EVM to the ATLYS FPGA Board. The two 80 pin connectors of the EVM and the 40-pin high speed expansion header of ATLYS Board are highly incompatible. So, as pointed out above, a custom interconnection PCB is designed and produced, whose photo is given in Figure 4.3.

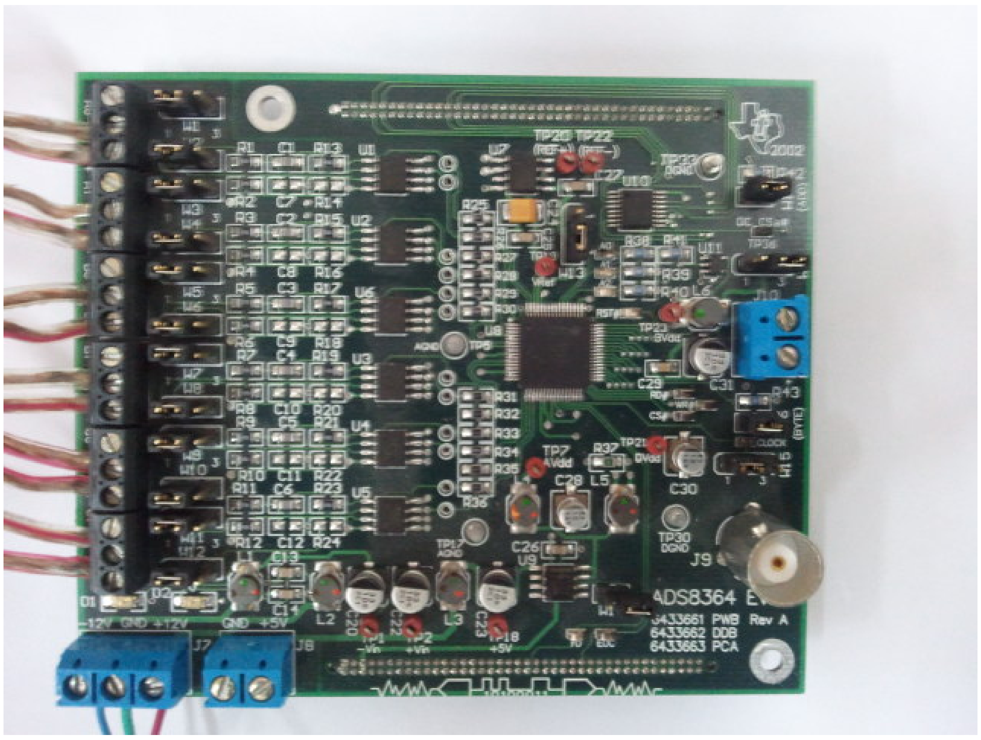


Figure 4.5: Top view of ADS8364 Evaluation Module

EVM is mainly composed of two parts: analog and digital. With the input buffers in the analog section, it provides level and impedance changes to the input signal. There is also a voltage reference circuit in the analog part. Digital part just connects the digital data interface of ADS8364 ADC to the 80 pin expansion connectors to be used by an external user, in our case, it is an FPGA. Using this EVM in the acoustic radar system is obviously very advantageous as it isolates the main system from the analog considerations of the ADC. It would take so much time to design a custom PCB for the ADC. The only disadvantage and challenge in using this EVM is to achieve the connection between that and ATLYS FPGA board.

4.2 Description of FPGA Building Blocks

In this section, the detailed description of the building blocks of the implemented radar system is given.

4.2.1 Transmit Pulse Generator Module

This module implements transmit delay-and-sum beamforming. The basic operation of Transmit Pulse Generator is generating (upon being triggered by Radar System Controller Module) six 24-Bit outputs whose phases are adjusted such that the intended steer angle is achieved.

Functional Description

The functional block diagram of the module is given in Figure 4.6. The operation of *Transmit Pulse Generator* Module is controlled by a finite state machine (FSM). After power-up and reset release, FSM is in idle mode, observing a possible write to the “steer angle register“. The input interface of this module is a WishBone Slave Port [23]. WishBone is an open source interface standard between IP Cores developed by OpenCores Organization. Using a standard data exchange protocol between modules rather than a custom interface for each module makes the system less complex, more flexible and eases the operation of the main processor, namely, *Acoustic Radar System Controller* module. Among two WishBone compatible IP Cores, one is Master and the other is Slave. Data exchange between Master and Slave takes place such that Master makes write/read accesses to the internal registers of the Slave. Slave cannot initiate a data exchange, only Master can.

Transmit Pulse Gen module is a WishBone Slave, which implies that there are some internal registers which are write/read accessible by a WishBone Master. Actually, there is only one internal register inside the module, which is *steer angle register*. WishBone Master (in our particular application, it is *Radar System Controller Module*) simply writes to the *steer angle register* via the WishBone port when it intends to transmit a pulse. The information written to the steer angle register is, as the name suggests, the angle information to which the transmit

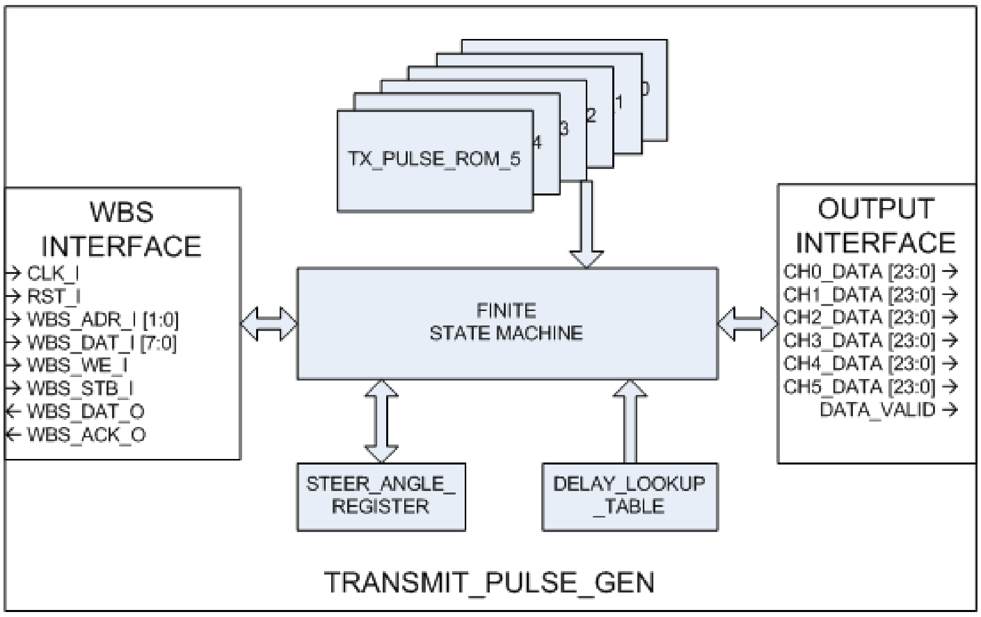


Figure 4.6: Tranmit Pulse Generator Functional Block Diagram

array is steered. Steer angle register may take values [0, 180] with a resolution of 1 degree. Once the steer angle register is written by the Acoustic Radar System Controller module, the FSM is triggered and the pulse outputting process starts.

There are six read only memories (ROMs, generated using XILINX Distributed Memory Generator [24]) inside the Transmit Pulse Generator module to which the transmitted pulse samples are recorded. The content of all the ROMs are just the same, they are all filled with acoustic radar pulse.

After triggered by a WishBone write access to the steer angle register, The Transmit Pulse Generator FSM starts to access channel ROMs. Its function is just reading the recorded radar pulse samples from the ROMs with a correct phase difference (which is dependent on the steer angle) and supplying them to the following modules with a "data valid" signal. The radar pulse recorded in ROMs is given in Section 3.1.7.

Note that delay-and-sum transmit beamforming is nothing but feeding the transmitter array elements with proper delays. There is a lookup table in the module called "delay lookup table" which holds the information of the delay per sensor for a specific steer angle in terms of number of samples. Note that for a specific steer angle, the phase difference between transmitter array element pulses is fixed, it is

$$\tau_k(\theta) = \\ = round\left(\frac{d \times \frac{cos(\theta)}{c}}{\left(\frac{1}{f_s}\right)}\right) (in\ number\ of\ samples) \\ k = 1, ..., 6 \tag{4.1}$$

where $\tau_k(\theta)$ is the delay for each sensor to steer the wave towards θ angle, d is the distance between array elements (in meters), θ is the steer angle (in radians), c is the speed of sound in air (in m/sec) and 1/fs is the sampling period (in seconds). Rounding operation in the formula is obvious as digital implementation necessitates integer (number of samples) delay values.

So, delay lookup table has 181 entries (starting from 0 degrees to 180 degrees) each of which holds the delay per sensor information for the corresponding steer angle. The content of the delay per sensor lookup table is given below in Figure 4.7

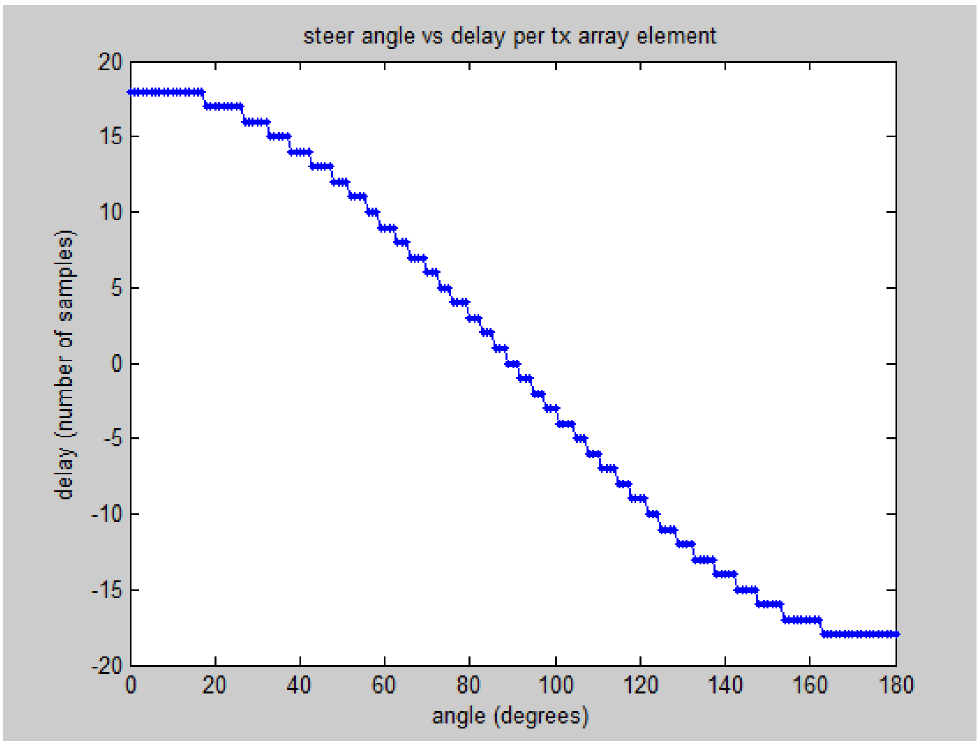


Figure 4.7: Delay Lookup Table Content

So, with a given steer angle information, FSM reads channel ROMs using the delay information and outputs the waveforms for each channel with a “data valid“ qualification flag.

The port description of the module is given in Appendix A.1

4.2.2 PCM1602 DAC Controller Module

This module is used to drive D/A converter, which is a Texas Instruments PCM1602, 24-Bit, 192-kHz Sampling, 6 Channel, Enhanced Multilevel, Delta-Sigma Digital-to-Analog Converter [27].

Functional Description

The module has 6 internal FIFOs (First-In-First-Out) which are used for buffering the incoming data stream coming from 6 channels (Figure 4.8). They are implemented using embedded Block RAM resources in Spartan-6 FPGA [25]. To generate FIFOs using those embedded Block RAM resources, software called “The XILINX LogiCORE IP FIFO Generator v8.2 [26]“ is used. LogiCORE IP FIFO Generator is a software which provides the user the ability of generating FIFOs with custom implementation choices including width, depth, status flags, memory type and the write/read port aspect ratios. It provides an optimized solution for all FIFO configurations and delivers maximum performance while utilizing minimum resources.

The FIFOs are asynchronous and have non-symmetric aspect ratios.

The fact that FIFOs are asynchronous implies that they have different write and read clocks. That kind of clock domain difference is implemented for the reason of isolating the upper layer controller from DAC Audio Serial Interface timing considerations. In other words, the master, which is Transmit Pulse Generator in the particular implementation, supplies data to DAC Controller module with its very own clock and has nothing to do with the DAC serial interface clock domain. So, this kind of clock domain isolation makes the system more modular, a change in upper layer system clock domain does not affect the DAC serial interface operations.

The fact that FIFOs have non-symmetric aspect ratios implies that they have different write and read data widths. In the particular implementation, the data written to the module is 24 bits (this is nothing but the output of the Transmit Pulse Generator module), the output of the FIFOs are 1 bit. This is because the DAC control interface requires serial communication, i.e.

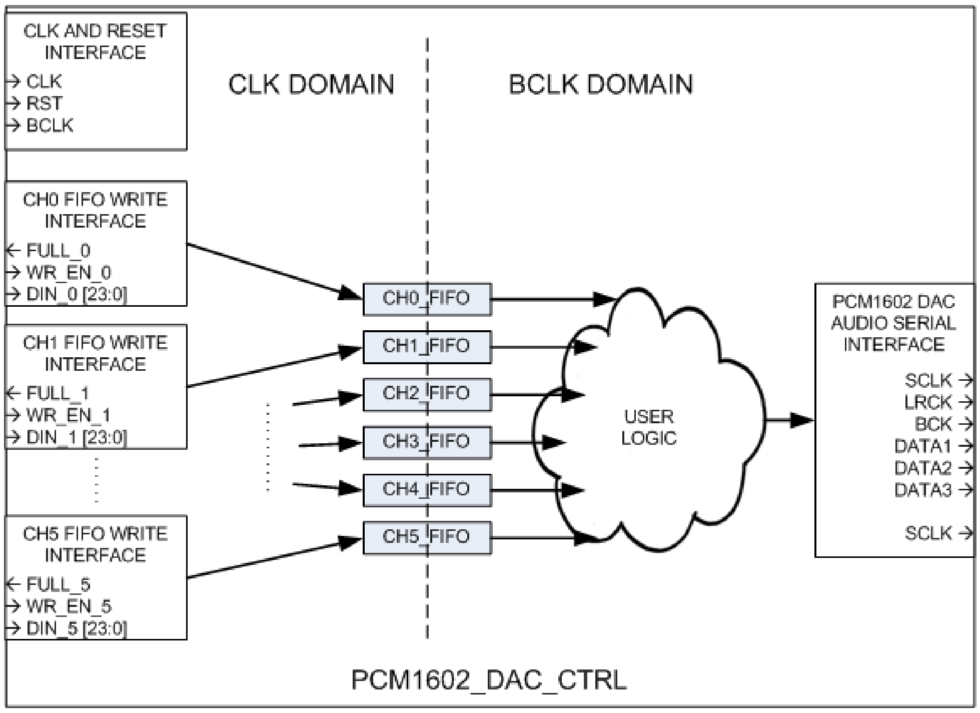
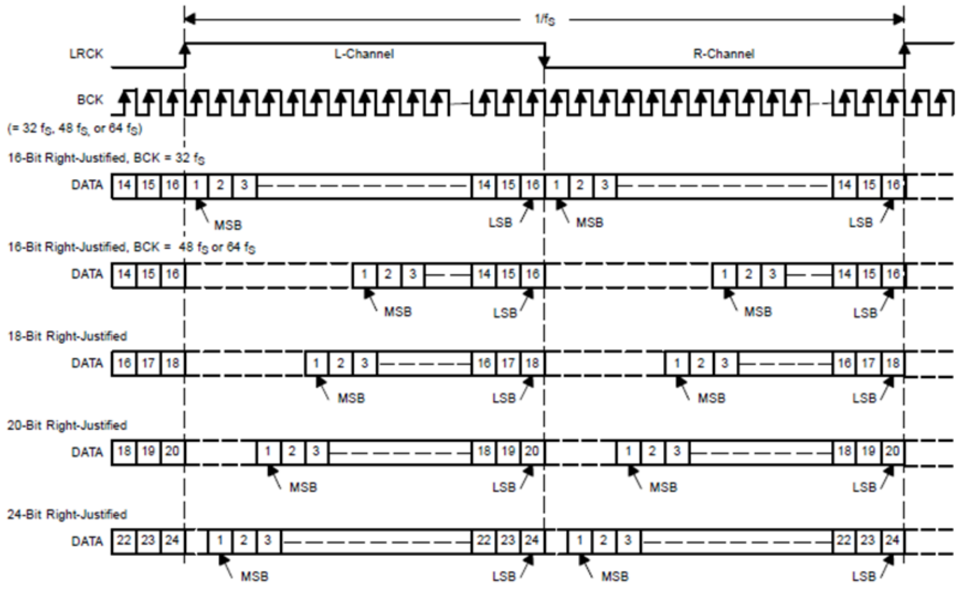


Figure 4.8: PCM1602 DAC Controller Functional Block Diagram

data is supplied to the DAC as a bit stream.

After this few words about the FIFOs inside DAC Controller, it is time to explain the operation of the module. Let us call the intelligence in the module as User Logic (as seen from Figure 4.9). User Logic mainly consists of 6 shift registers, each of which is of 24 bits. This shift registers are shifted into the DAC audio serial interface.

DAC Audio Serial Interface is a 5-wire synchronous serial port, which description is given in Table A.2. It supports industry-standard audio data formats, which are given in Figure 4.9. Among those standard data formats, 24-Bit Right-Justified (which is default format of PCM1602, actually) format is used. In our implementation, the sampling frequency is chosen as 52083. It is obviously far beyond satisfying the Nyquist sampling rate as the frequency of the transmitted pulse is 1417 Hz. In choosing the sampling frequency, some considerations regarding implementation ease have been taken into account, which is about the clock generation. As explained in Section 4.2.8, using a DCM, it is so practical to generate a 2.5 MHz clock for the BCK from the on-board 100 MHz oscillator on the ATLYS FPGA Board, whose functional block diagram is given in Figure 4.2. So, this is why sampling frequency f_s is



1/fS
LRCK
L-Channel
R-Channel
BCK
(= 32 fS, 48 fS, or 64 fS)
16-Bit Right-Justified, BCK = 32 fS
DATA
MSB
LSB
16-Bit Right-Justified, BCK = 48 fS or 64 fS
18-Bit Right-Justified
20-Bit Right-Justified
24-Bit Right-Justified

Figure 4.9: Standard Audio Data Input Formats

52083 Hz (In 24-Bit Right-Justified audio data format f_s = BCK/48 = 2.5 MHz/48 = 52083).

Back to the User Logic explanation, it continuously drives the DAC Audio Serial Interface no matter there is data present in the input FIFOs or not, i.e. 6 channel shift registers are continuously shifted into the DAC Serial Interface. If the input FIFOs are empty, their values are all zero, so DAC is fed with zeros. When there is some data in “all“ the FIFOs, they are all read and the channel shift registers are updated with the data read from channel FIFOs. The keyword here is “there is some data in all the FIFOs“. It is important, because, the FIFOs must be read synchronously, in other words, they must be read all together, this is vital to keep the phase relation between all 6 channels.

The port description of the module is given in Appendix A.2

4.2.3 ADS8364 ADC Controller Module

This module is designed to capture data from the 250 kSPS, 16 Bit, 6-Channel Simultaneous Sampling Analog-to-Digital Converter [32].

Functional Description

After power up, ADC Controller Module starts to capture data from the ADS8364 ADC continuously. ADS8364 includes six, 16-bit, 250 kSPS ADCs.

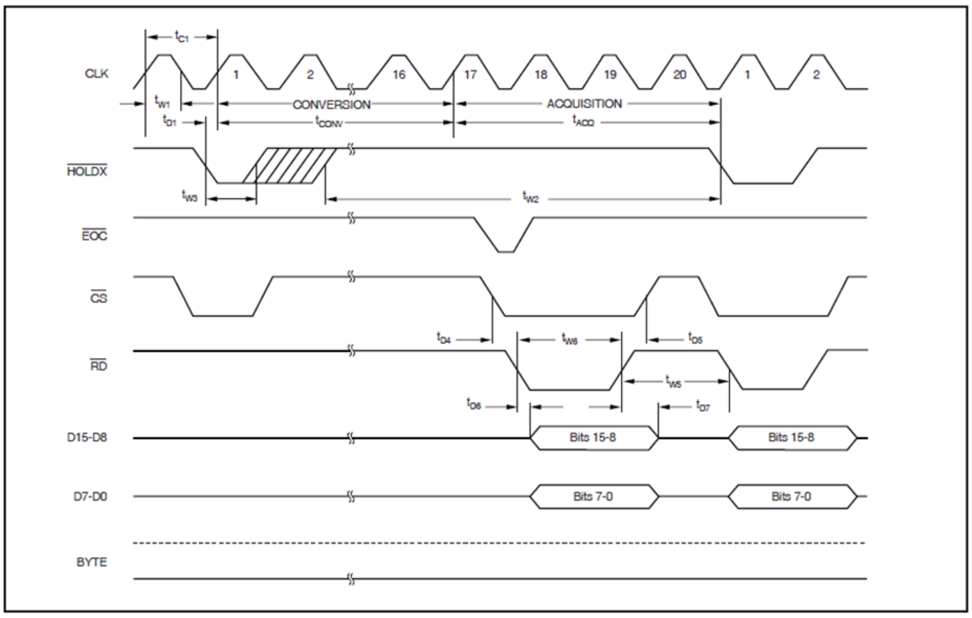


Figure 4.10: ADS8364 Data Capture Timing Diagram

The data capture process relies on the timing given in Figure 4.10: There are three hold signals (HOLD_A_N, HOLD_B_N and HOLD_C_N) which initiate the conversion on the specific channel pair. The User Logic in the module just asserts all HOLD_N signals to initiate a conversion. After 16.5 clock cycles after initiating a conversion, ADC asserts EOC (End Of Conversion) signal which implies that capture is over and data is present in the internal registers of ADS8364. After that, User Logic performs read accesses to the ADS8364 registers via asserting RD signal. It reads channel data in order, first channel 0 data, then channel 1 and channel 5 is the last one. The data read from ADS8364 is supplied to the output immediately via a FIFO write interface. This module is designed to be used with FIFOs following it, as seen from Figure 4.11.

ADC Controller Module uses ADS8364 in the maximum sampling mode which is 250 kSPS. To achieve this, it supplies the ADC with a 5 MHz clock.

The port description of the module is given in Appendix A.3

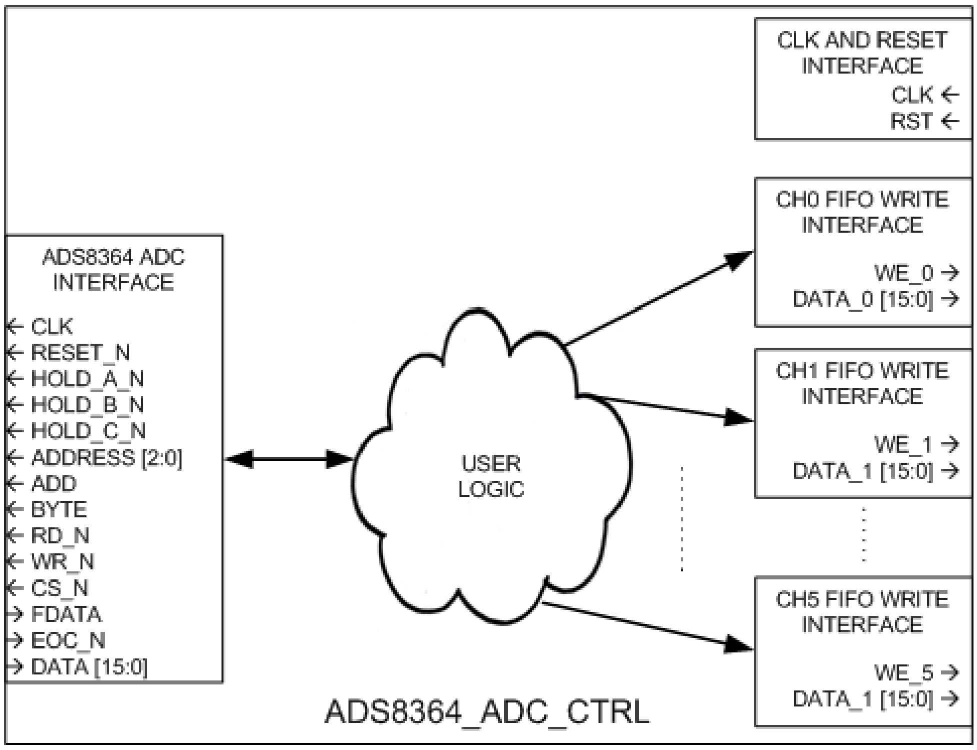


Figure 4.11: ADS8364 Data Capture Timing Diagram

4.2.4 DC Offset Remover

Due to the physical nature of the microphones and Analog-to-Digital Converter circuit used in the system, which is explained in Section 4.1.3, the captured digital waveform has a very high DC Offset. Before further processing the waveform, i.e. low-pass filtering, delay-and-sum beamforming and then matched filtering, the removal of the dc offset is necessary. To achieve the DC offset removal, an RC circuit is digitally implemented [29].

In analog world, the easiest way to find the DC offset of a waveform is obviously an RC circuit, which is given in Figure 4.12.

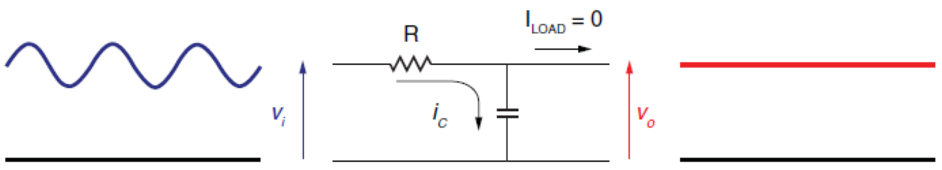


Figure 4.12: Analog RC Circuit to Determine the DC Offset of a Signal

The circuit can be analyzed as follows. The current flowing through the capacitor is determined by the voltage across the resistor R,

$$i_c = \frac{v_i - v_o}{R} \tag{4.2}$$

The charge Q on a capacitor is defined as

$$Q = C \times V = I \times T \tag{4.3}$$

So, by (4.3), for a period time T, for a constant current I, the voltage accross the capacitor rises by,

$$V = \frac{I \times T}{C} \tag{4.4}$$

Combining equations (4.2) and (4.4), the voltage rise accross the capacitor during a period of time ΔT can be expressed as

$$\Delta V_o = \frac{\Delta T}{R \times C} \times (v_i - v_o) \tag{4.5}$$

By using a constant k, which is defined as $k = \Delta T/(R \times C)$, the output voltage becomes

$$V_o^{'} = V_o + k \times (v_i - v_o) \tag{4.6}$$

at the end of each period ΔT. For example, consider a sinusoidal waveform with a frequency of 2 KHz. Let's choose k as 0.01. The signal itself and the DC offset value calculated by (4.6) is given in the following Figure 4.13. This example is generated by a matlab script. The ΔT value is chosen as 1/100 th of the period of the sinusoidal signal, which is 5 usec ($f = 2\ KHz \rightarrow T = \frac{1}{f} = 0.5\ msec \rightarrow \Delta T = \frac{T}{100} = 5\ usec$). So the corresponding RC coefficient is chosen as 0.25 msec ($RC = \frac{\Delta T}{k} = \frac{5\times10^{-3}}{0.02} = 0.25 \times 10^{-3}$)

The *DC Offset Remover* used in the Acoustic Radar System is a digital implementation of the analog RC circuit analyzed above. Its structure is given in Figure 4.14. Six of them are employed in the system, one for each input channel. k is selected as small as possible (it is selected as $\frac{1}{2^{15}} = 000000000000001_2$) to obtain the most smooth output waveform (Note that large RC values result in smaller k values and large RC implies a more smooth voltage waveform across the capacitor). Note that, after power-up, the calculated DC offset value is just 0 and it takes some time until the real offset is calculated (remember the charge-of-a-capacitor analogy), this time increases as k value decreases.But this initial delay is not a problem for our system, obviously, because this time would already pass in the initial idle phase of the system.

The port description of the module is given in Appendix A.4

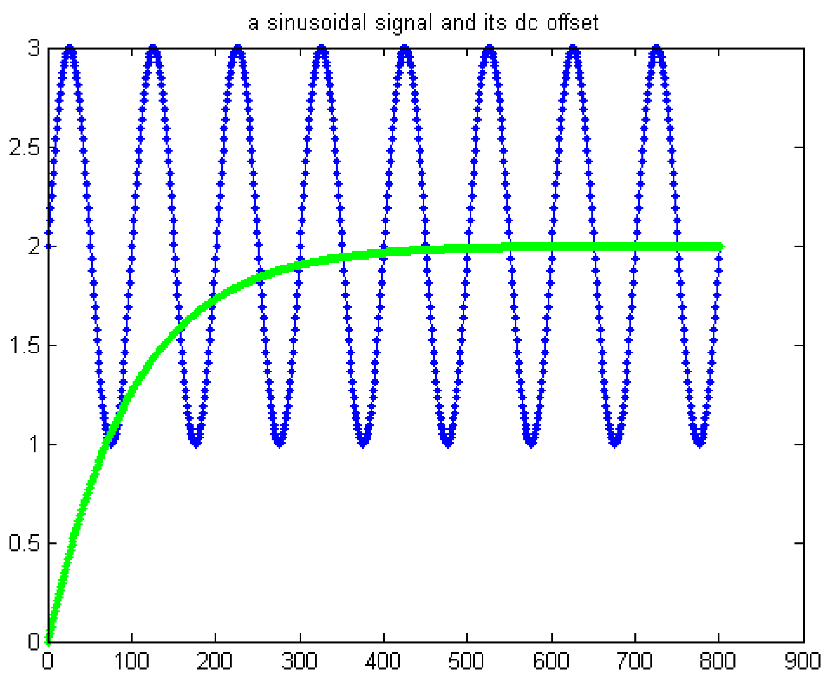


Figure 4.13: DC offset of a sinusoidal signal calculated by the RC Circuit

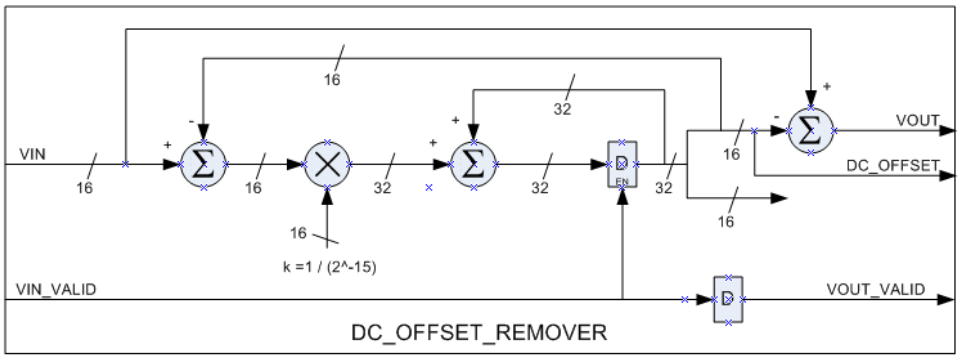


Figure 4.14: *DC Offset Remover*, a digital implementation of the RC Circuit

4.2.5 6-Channel FIR Low-Pass Filter Module

The data captured from ADCs has considerable amount of high frequency components due to physical conditions. These high frequency components result in a noisy signal which makes further processing more difficult. So, before applying beamforming and matched filtering, those high frequency components must be filtered out from the captured waveforms. In order to clean the signal from those noisy components, an obvious method is low-pass filtering.

Matlab filter specification object is used to design the filter [30].The specifications of designed low-pass filter is given below in Table 4.1. In the following Figure 4.15, the magnitude and phase response of the low-pass filter is also given.

Response	Lowpass
Fs (sampling frequency)	250000
Filter Order	32
Cutoff Frequency	2000
Filter Structure	Direct-Form FIR
Design Algorithm	Window-Hamming

Table 4.1: Implemented Low-Pass Filter Specifications

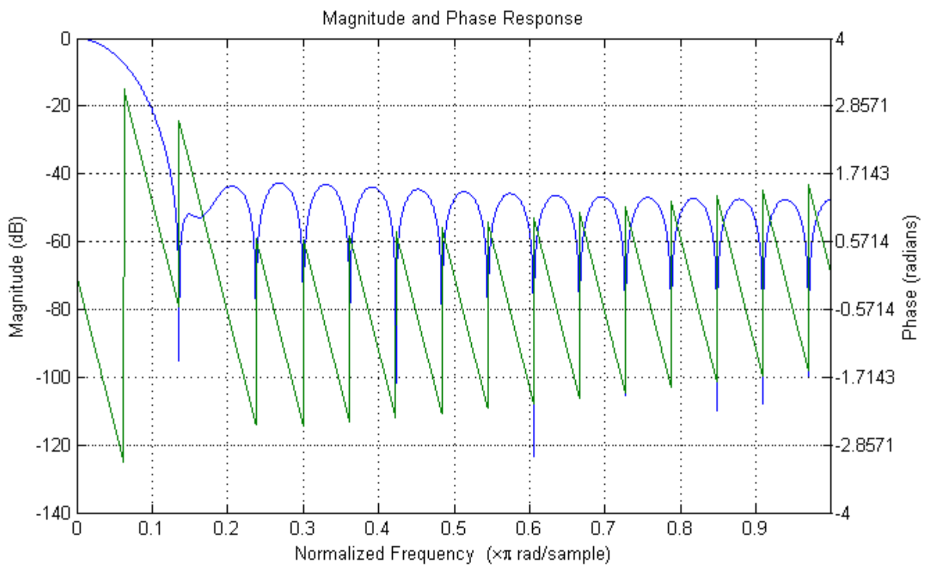


Figure 4.15: Frequency Response of the Implemented Low-Pass Filter

In the time-domain, a causal FIR filter of order N is characterized by the following input-output relation:

$$y[n] = \sum_{k=0}^{N} c[k]x[n-k] \tag{4.7}$$

In FPGA implementation of the low-pass FIR filter whose specifications is given above in Table 4.1, Direct-Form Structure is used. But, the direct form structure is not used "directly" because of some implementational aspects, namely, the timing requirements in FPGA. A direct form realization of the FIR filter structure, which is named as "canonical" form is given below in Figure 4.16.

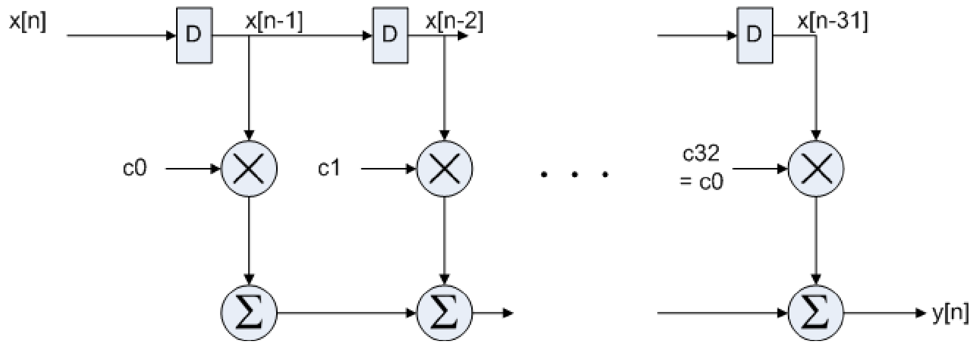


Figure 4.16: A direct form realization of an FIR filter (1 channel)

Implementing the canonical filter structure given in Figure 4.16 directly on FPGA would cause some timing problems. Note that the structure includes many multipliers and adders, which are combinatorial elements. In the mentioned structure, those elements are directly connected, i.e. no delay elements (registers) are employed between them. This structure would result in poor timing performance when implemented on FPGA. A pipelined structure which improves the timing performance and obviously the maximum operation frequency is given below in Figure 4.17. The delay elements highlighted by green color in the figure are inserted in order to improve timing.

The FIR filter structures given above are all for one channel only. Note that in the acoustic radar system, there exists 6 input channels and they, all, has to be filtered. An obvious approach to achieve low-pass filtering of 6 channels would be to employ 6 FIR filter structures, one for each channel. But, note that this would result in an excessive usage of FPGA resources. Moreover, to improve timing performance and source utilization, a special structure

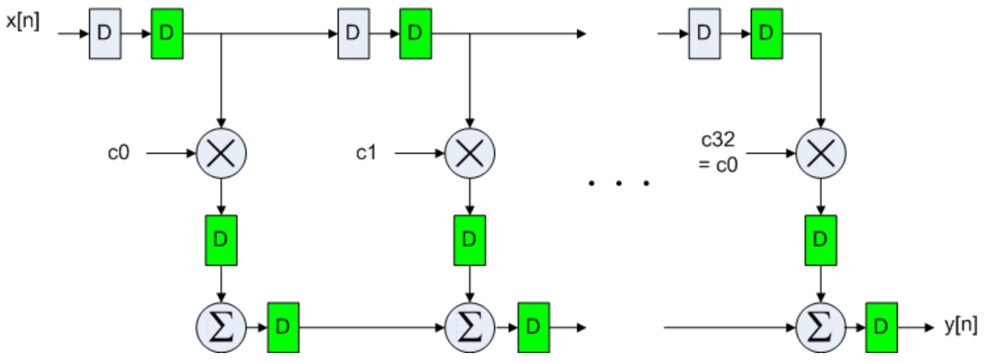


Figure 4.17: Pipelined structure of the FIR filter to improve timing (1 channel)

in the FPGA which is called “DSP48 slice“ (it is to be explained in detail shortly) is used to implement FIR taps. One FIR filter structure consumes 33 DSP48 primitives, one for each tap, so to employ a FIR filter for each channel, $33 \times 6 = 192$ DSP48 primitive would be used. But, there are only 58 DSP48 slices in Spartan-6 (XC6SLX45) FPGA [19]. So, an alternative implementation is used in order to filter 6-channel data using only one FIR structure. It is given in Figure 4.18.

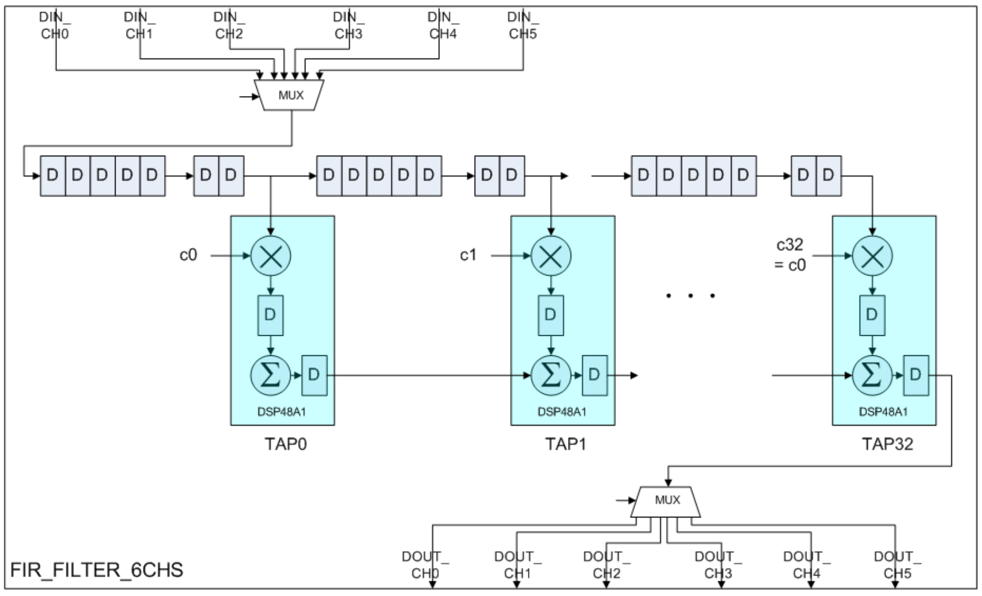


Figure 4.18: 6-channel low-pass FIR filter structure

The structure given in Figure 4.18 is derived from that of pipelined one in Figure 4.17 by adding 5 more delay elements to input line, i.e. between taps before the multipliers. Channel data is fed to the filter structure in a regular order starting from ch0 data to ch5 data. Similarly, the filtered data is fed to the output of the module in turns. The ordering of the input and output data is achieved by input and output multiplexers, respectively.

Note from Figure 4.18 that, to implement the filter taps, a special FPGA structure, which is called *DSP48 Slice* [31], is used. DSP48 slices are dedicated primitives in some Xilinx FPGAs which support many DSP algorithms with *minimal use of the general purpose FPGA fabric*. Note from Figure 4.19 that, at first look, the DPS48 slice contains an 18-bit pre-adder followed by an 18x18 bit two's complement multiplier and a 48 bit sign-extended adder/subtracter/accumulator that is widely used in digital signal processing.

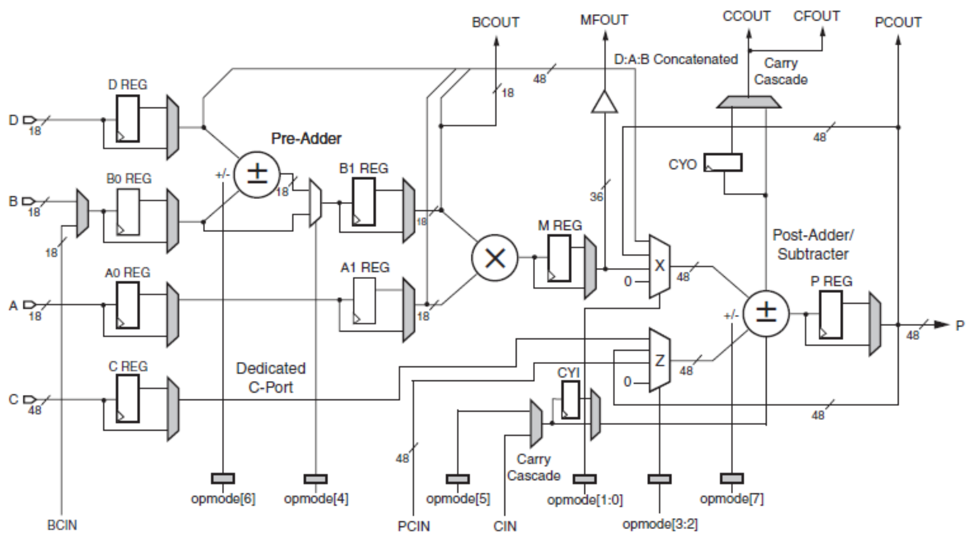


Figure 4.19: DSP48A1 Slice

In Figure 4.20, the parts of DPS48 Slice used in our particular implementation is highlighted.

The port description of the module is given in Appendix A.5

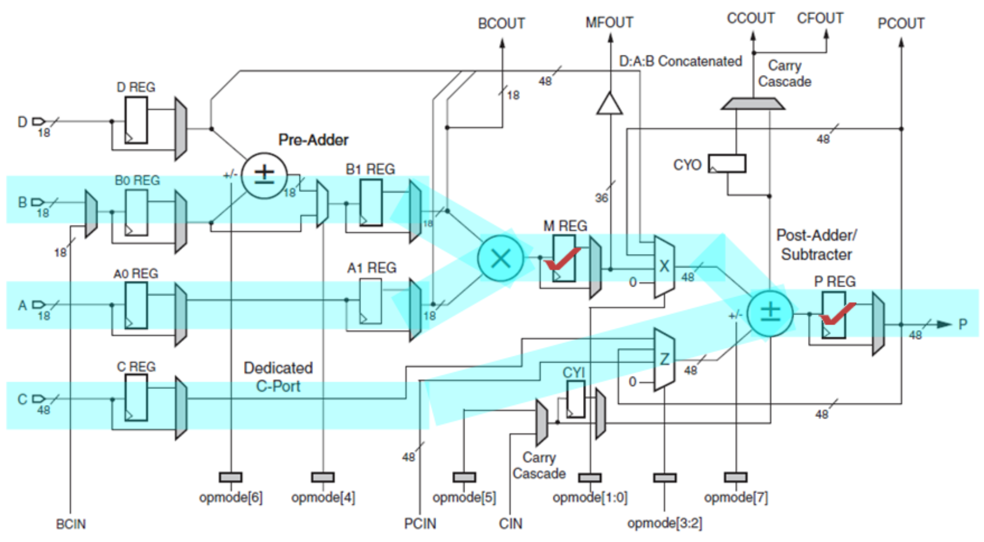


Figure 4.20: DSP48A1 Slice, used parts highlighted

4.2.6 Receive Delay and Sum Beamformer Module

This module implements the well-known delay and sum beamformer digitally. In addition to beamforming, it is used in *phase calibration* of channel waveforms as well.

Delay-and-Sum Beamforming is the simplest implementation of a beamformer. The output of a Delay-and-Sum beamformer can be expressed as

$$y[n] = \sum_{i=1}^{L} x_i \left[n - \Delta_i\right] \tag{4.8}$$

which is simply delaying input channel data “correspondingly“ and then sum them. L is the number of input channels, which is 6 in our case; Δ_i is the delay amount for the ith channel. The delay amount for the channels depends on the steer angle of the beamformer.

Functional Description

The operation of the module is controlled by a Finite State Machine, as shown in Figure 4.21. The flow chart of the state machine is given in Figure 4.22. As soon as reset is released, FSM observes WBM port (*wait_start_trigger*) for a probable write access to the *steer angle register*. The operation of the FSM is triggered by a write access to this register. After a successfull write to *steer angle register*, the first action that FSM takes is to perform phase calibration on channels (*perform_calibration*).

Phase calibration is a process in which channel waveforms are adjusted by applying predetermined initial delays before further processing. The delay amounts which are to be applied on channels are parametric, they can be set by the Acoustic Radar System Controller user via graphical user interface mentioned in Section 3.1.4. There are six wishbone accessible internal calibration registers in beamformer module, which are namely, *ch0_calibration_offset_reg, ..., ch5_calibration_offset_reg*. Each of them holds the calibration delay value for the corresponding channel. The delay values which are to be applied to the channels are calculated experimentally by the system user and then set from the GUI.

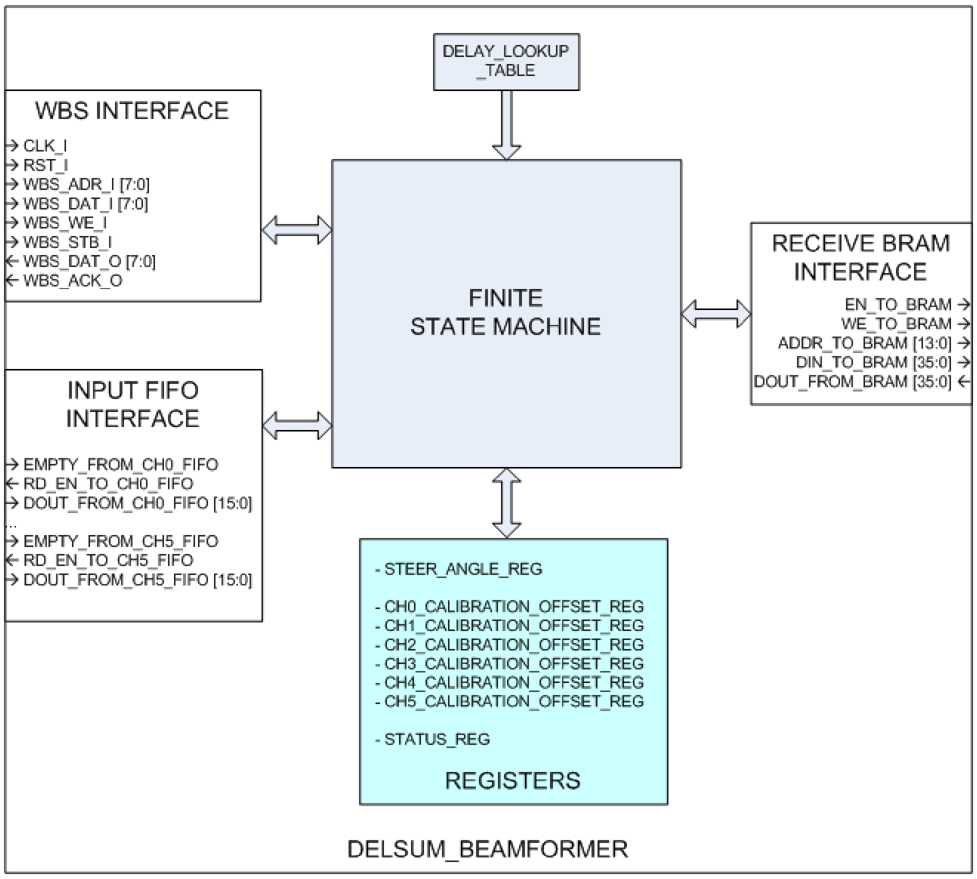


Figure 4.21: Receiver Delay-and-Sum Beamformer Functional Block Diagram

After channel phase calibration process is over, the primary task of the module, which is delay-and-sum beamforming, starts. First of all, the channel delay values which should be applied to the channels for the current steer angle (this information is hold by *steer_angle_reg*) are calculated (*update_channel_delays*) using the information in *delay lookup table*. *Delay* *lookup table* is a read-only-memory with 180 entries, each of which holds the information of the delay amount between adjacent channels for the corresponding steer angle. As soon as channel delay values are updated (it just takes two clock cycles), the adjustment of channels starts (*adjust_channels*). In this state, necessary number of samples are read from each channel FIFOs and thrown away. After this process, what is achieved is that all channel waveforms captured from the current steer angle become in-phase. So, at this point, having applied the corresponding delays to the channel waveforms, the next action to take is to sum those delayed or phase-adjusted waveforms.

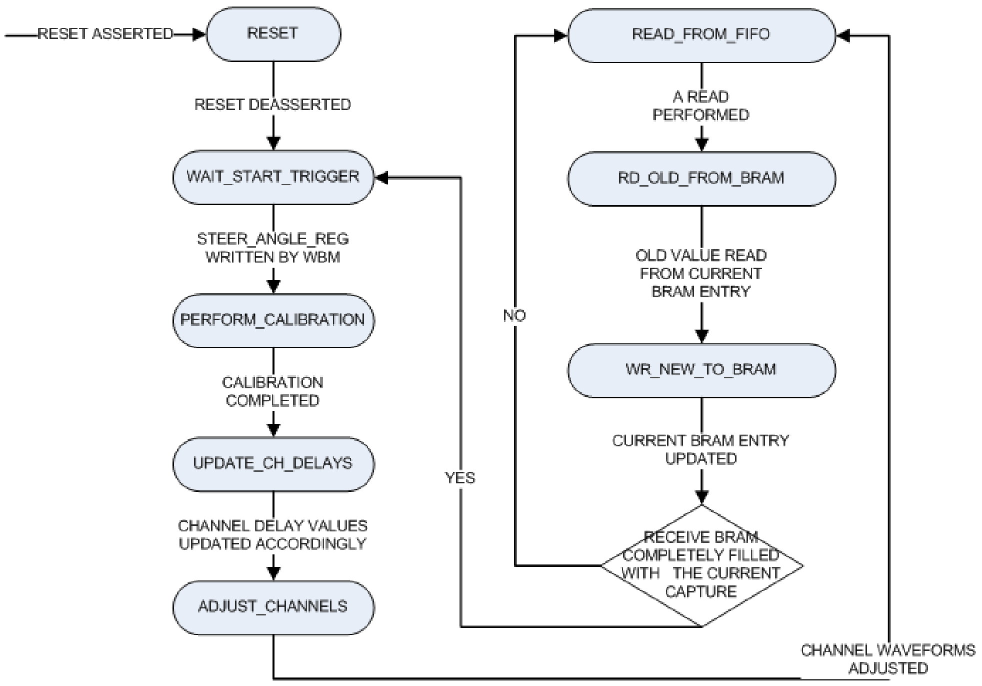

RESET ASSERTED
RESET
RESET DEASSERTED
WAIT_START_TRIGGER
STEER_ANGLE_REG WRITTEN BY WBM
PERFORM_CALIBRATION
CALIBRATION COMPLETED
UPDATE_CH_DELAYS
CHANNEL DELAY VALUES UPDATED ACCORDINGLY
ADJUST_CHANNELS
READ_FROM_FIFO
A READ PERFORMED
RD_OLD_FROM_BRAM
OLD VALUE READ FROM CURRENT BRAM ENTRY
WR_NEW_TO_BRAM
CURRENT BRAM ENTRY UPDATED
RECEIVE BRAM COMPLETELY FILLED WITH THE CURRENT CAPTURE
NO
YES
CHANNEL WAVEFORMS ADJUSTED

Figure 4.22: Receiver Delay-and-Sum Beamformer Finite State Machine Flow Diagram

Delay and sum beamforming process consists of three states; namely, *read_from_fifo*, *rd_old_from_bram* and *wr_new_to_bram* states. After adjustment of the channels is over (the procedure is explained above), the only action to take is - for a conventional beamformer - just to sum all data read from channel FIFOs. But, in *Receive Delay-and-Sum Beamformer Module*, there is one extra feature: it provides the user (the user is *Acoustic Radar System Controller* for the implemented system) the ability to apply beamforming over multiple captures. What it means may be explained as follows. When the user runs the module, the output is recorded to a RAM, which is *Receive BRAM* (BRAM: Block RAM). When the module is run again without clearing the RAM content, the previous beamforming output is not overwritten, the current output is written to the BRAM after summed with the previous one. This feature is added to the beamformer module to be able to analyze the effect of beamforming performed over multiple captures on the decision process. The “sum“ portion of delay-and-sum beamforming is achieved in three steps as mentioned above, a read is performed from all of the channel fifos (*read_from_fifo*), these six samples read from channel FIFOs are summed. This value is nothing but the beamformer output for the current channel samples. Next, the previous value is read from the RAM (*read_old_from_bram*), this is summed with the current one and then written to the RAM (*wr_new_to_bram*). If the last sample is not yet reached, the procedure is repeated starting from *read_from_fifo*. If the last sample of the beamformer is calculated, the module returns to its initial state -which is *wait_start_trigger*- to observe WBM port for probable requests.

The port description of the module is given in Appendix A.6

4.2.7 Correlator Module

Correlation of the beamformer output with the transmitted pulse is performed by this module. The module has the ability to calculate the correlation with a user defined decimation (the user can change the “decimation factor“ of the correlator using *System Configuration and Display Unit* GUI). The correlator output is given as,

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n+k] \tag{4.9}$$

In (4.9), h[n] is the nothing but the transmitted acoustic radar pulse, N is the length of this pulse, x[n] is the received waveform (beamformer output in our case). Note that the above equation can be implemented as an FIR filter whose coefficients are the samples of h[n]. In our implementation, N = 500, i.e. transmitted radar pulse has 500 samples. (note that the duration of the radar pulse is 2 msec. The receive sampling rate of the system is $f_s = 250$ KSPS, or the sampling period $\mathrm{T}_s = \frac{1}{f_s} = 4\,\mu$sec. So, the number of samples in our radar pulse is 500 samples) Obviously, implementing an FIR filter with such an extremely high number of taps in the FPGA is either impossible or results in an excessive use of resources. In our case, actually, it is impossible, becuse there is not enough resources in the FPGA.

So, rather than implementing the correlator simply with a conventional FIR filter structure, a different technique is used to calculate the correlation. In the designed structure, *only a single multiply-accumulator* is employed (this multiply-accumulator may be thougth as a tap of an FIR filter). The accumulator structure is given in Figure 4.23. (4.9) is implemented using this circuit in the following way: For a particular n (note the “i“ subscripts in Figure 4.23), the circuit is run for 500 (N = 500) clock cycles. So, after 500 clock cycles, the result is nothing but the correlation for this particular n. Then, the next correlation sample is calculated in another 500 clock cycles. So, total time required to calculate all the correlation samples is

$$\begin{aligned} T_{correlation\ time} &= correlation\_length \times N \times T_{op} \\ &= (N + capture\_length) \times N \times T_{op} \\ &= (500 + 14112) \times 500 \times 10\ nsec \\ &= 0.7306\ sec \end{aligned} \tag{4.10}$$

In (4.10), T_{op} is the operating period of the acoustic radar system (note from Figure 4.1 that system operates in 100 MHz). Note that by using this structure, 500 times less resources are used, but the speed in decreased drastically in return, the time spent to calculate the correlation for only a single channel is about 1 seconds. This is the time passed when no decimation is employed in the calculation process. The time required to calculate the correlation decreases with the square of the decimation factor used. For example, if a decimation factor of 16 is used, the time required to calculate the correlation would be $0.7306/16^2$ = 2.9 msec, which is obviously a much more acceptible value than 0.7306 seconds. So, if fast processing is required, the correlator should be used with a decimation factor which is different from “1“.

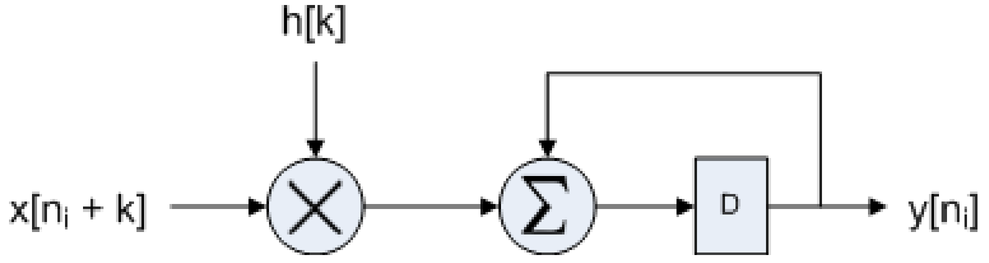


Figure 4.23: Multiply-Accumulate circuit which used to calculate a correlation sample

Functional Description

The *correlator* module is basicly formed by a DSP48 primitive, a read-only-memory in which the radar pulse is recorded and a finite state machine, as shown in Figure 4.24. The multiply-accumulator circuit in Figure 4.23 is implemented using a DSP48 slice [31] in the FPGA (refer Section 4.2.5 for an explanation of the DSP48 slice). The operation of the module is controlled by the internal finite state machine, whose flow is given in Figure 4.25.

As soon as reset is deasserted, module starts to observe WBM port for a probable write access to *corr_dec_factor_reg* register (*wait_start_trigger*). When a valid write access is performed to this register by WBM, the FSM is triggered and starts correlation calculation operation with the desired decimation factor. As explained above in detail, a particular sample of the correlation is calculated in a cycle which lasts for 500 clock periods (when decimation is used, it takes 500/decimation_factor). FSM runs the multiply-accumulator DSP48 (Figure 4.24) to calculate the particular sample of the correlation (*mult_acc_cycle*). The inputs to the multiply-accumulator are, as explained above, the radar pulse (it is recorded in a ROM) and the beamformer output (it is recorded in the Receive BRAM, section 4.2.6). After a particular sample of the correlation is calculated, i.e. after a multiply-accumulate cycle is over, FSM clears multiply-accumulator DSP48, which is required before the next cycle (*reset_mult_acc_dsp48*). An important feature to point out is that correlator module uses the very same BRAM with the beamformer, which is the *Receive BRAM*, to record its output, which results in a great saving in FPGA resources. After all correlation samples are calculated, the next action that correlator module takes is to find the maximum value of the correlation and the index of that maximum value (*find_corr_max*). For this purpose, the entire Receive BRAM is scanned by the module and the results are recorded to the registers *corr_max_index_reg* and *corr_max_value_reg*, which are soon to be read by the *Acoustic Radar System Controller*.

The port description of the module is given in Appendix A.7

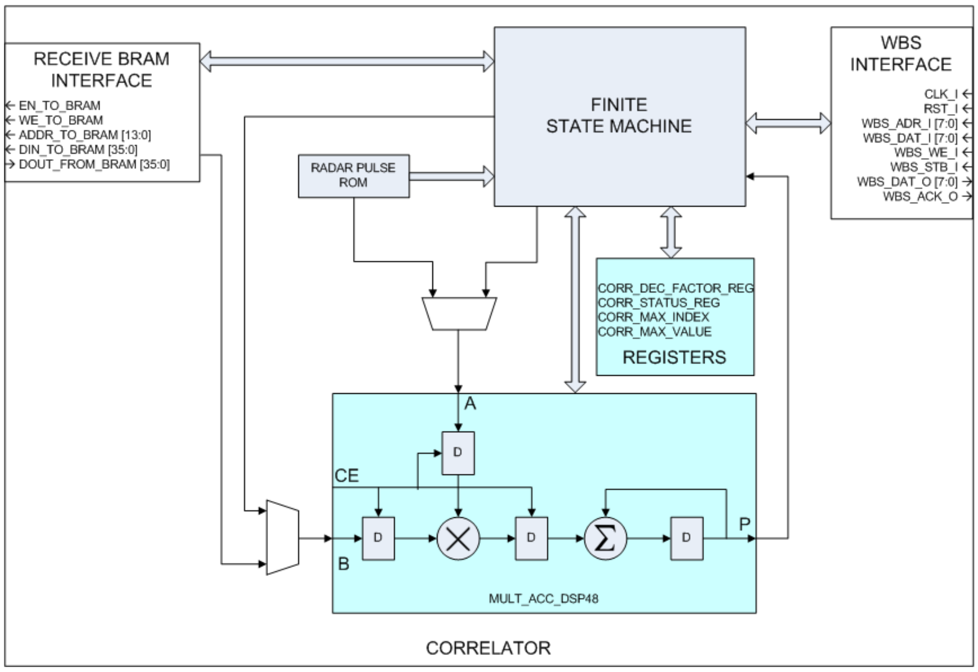


Figure 4.24: Correlator Functional Block Diagram

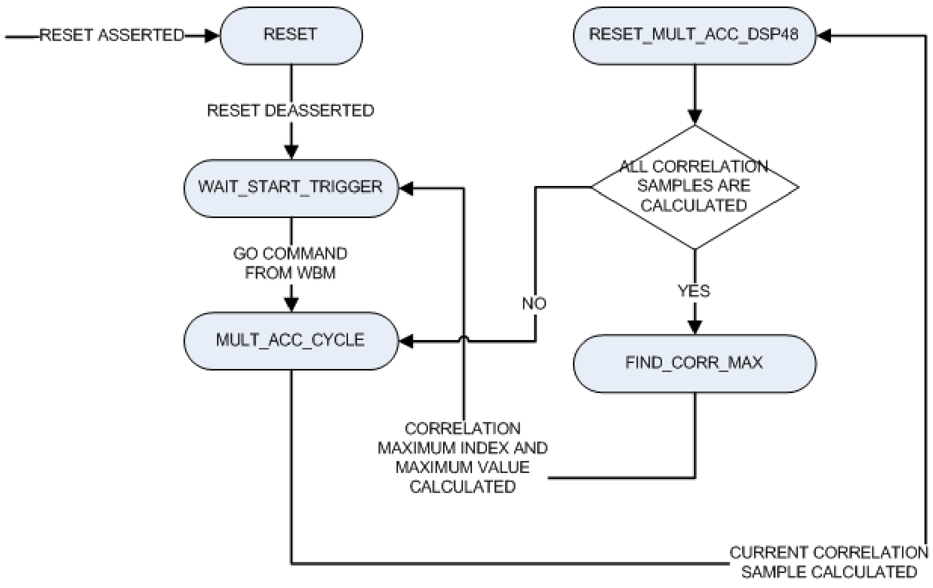

RESET ASSERTED
RESET
RESET DEASSERTED
WAIT_START_TRIGGER
GO COMMAND FROM WBM
MULT_ACC_CYCLE
RESET_MULT_ACC_DSP48
ALL CORRELATION SAMPLES ARE CALCULATED
NO
YES
FIND_CORR_MAX
CORRELATION MAXIMUM INDEX AND MAXIMUM VALUE CALCULATED
CURRENT CORRELATION SAMPLE CALCULATED

Figure 4.25: Correlator Finite State Machine Flow Diagram

4.2.8 Clock Generation Module

This module generates all the clock signals that acoustic radar system needs for its operation. System needs 100 MHz, 20 MHz and 5 MHz clock signals. Clock Generation module derives these clocks from the external 100 MHz clock input signal which is sourced from the 100 MHz CMOS oscillator mounted on ATLYS FPGA Board [20].

Functional Description

Clock Generation Module employs a PLL_BASE primitive for generating the required clocks [33]. PLL_BASE primitive provides access to the most frequent used features of a standalone PLL (Phased Locked Loop) circuit embedded in the Spartan-6 FPGA. The PLL is an embedded circuit which has the ability to phase shift, multiply and divide, modify the duty cycle and jitter filtering of an input clock.

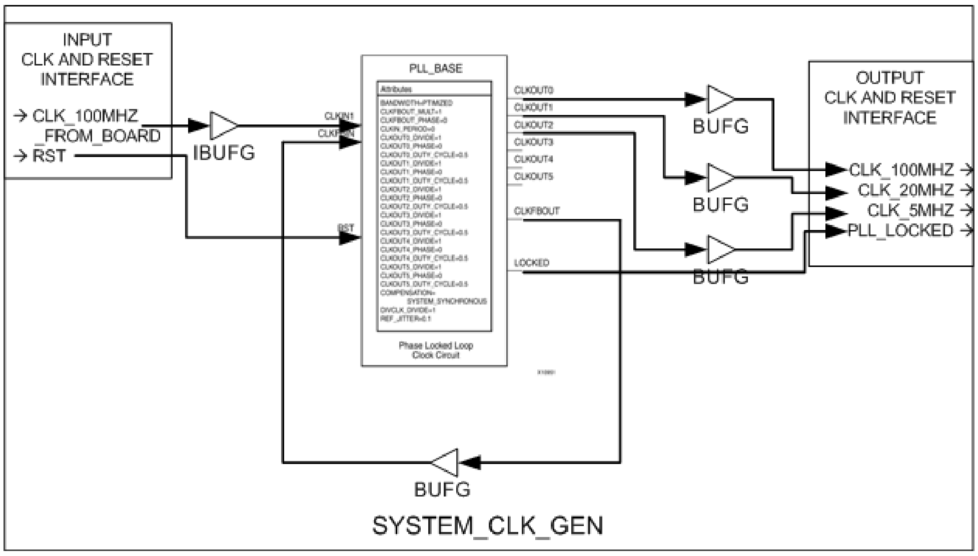


Figure 4.26: Clock Generation Module Functional Block Diagram

In the Clock Generation Module, the PLL_BASE primitive is configured in such a way that the input clock is divided by 1, 5 and 20, which results in the 100, 20 and 5 MHz clocks respectively. Moreover, this generated clocks are not arbitrary, they are synchronous to each other, i.e., phase aligned, which is very important for the design. 100 MHz clock is used by the upper layer controller in the system, which is Acoustic Radar System Controller Module. 20 MHz clock is fed to the DAC as the system clock input SCLK (see Table A.8). 5 MHz clock is supplied to DAC Controller module, it drives the DAC audio serial interface with this clock.

As obvious from Figure 4.26, the Clock Generator Module also employs BUFG primitives in addition to the PLL_BASE primitive. BUFG is a vital element in the FPGA Structure. This is “Global Clock Buffer“ that connects the input signal to global routing sources of the FPGA for low skew distribution of the signal. XILINX guarantees that the BUFG output signal will reach the clock inputs of all flip-flops within the fabric at almost exactly the same time. So, the output signals of Clock Generator Module may confidently be used as clock signals within the FPGA. (the IBUFG primitive in Figure 4.26 is the same as BUFG, the only difference is that it is located in the input pad of the FPGA, so this primitive may only be used for buffering external signals).

One more important signal that must be pointed out in this part is the “LOCKED“ output of PLL_BASE primitive. This is an active high asynchronous output from PLL which provides the user with an indication that the PLL has achieved phase alignment and is ready for operation. This signal is used as a global reset in the Acoustic Radar System.

The port description of the module is given in A.8

4.2.9 Ethernet Controller Module

The Computer (which is the Configuration and Display Unit of the system) and the FPGA (which is the main processor of the system) communicate over an Ethernet link via UDP (User Datagram Protocol) standard. *Ethernet Controller* module provides the main controller of the system, which is *Acoustic Radar System Controller* module, the ability to send/receive packets to/from the computer via Ethernet link without dealing with the protocol issues.

Functional Description

Ethernet Controller module is mainly formed by two parts, as seen from Figure 4.27, which are

1. Xilinx Soft Temac Locallink Unit and
2. ETH_CTRL_FSM Unit

Xilinx Soft Temac Locallink module is generated using the LogiCORE IP core generator software. It consists of Tri-Mode Ethernet MAC Soft IP Core and Transmit/Receive Locallink FIFOs [34]. Tri-Mode Ethernet MAC (TEMAC) IP Core is a soft IP [35], i.e. it is implemented using FPGA logical resources, it is not an embedded hard MAC IP Core (Spartan-6 FPGA does not have an embedded hard Ethernet MAC core, there are FPGAs, like Virtex-5, which have embedded hard Ethernet MAC Cores).

TEMAC Soft IP Core implements the Ethernet MAC protocol; in the particular implementation, it operates in full-duplex mode. There are two additional FIFOs also generated by LogiCORE IP Core Generator Software, which are transmit and receive LocalLink FIFOs. Those FIFOs provide the user (in our case, it is ETH_CTRL_FSM unit) the interface with the Ethernet MAC IP Core. They have a special interface signaling different from ordinary FIFOs, which is called "LocalLink Interface" [34].

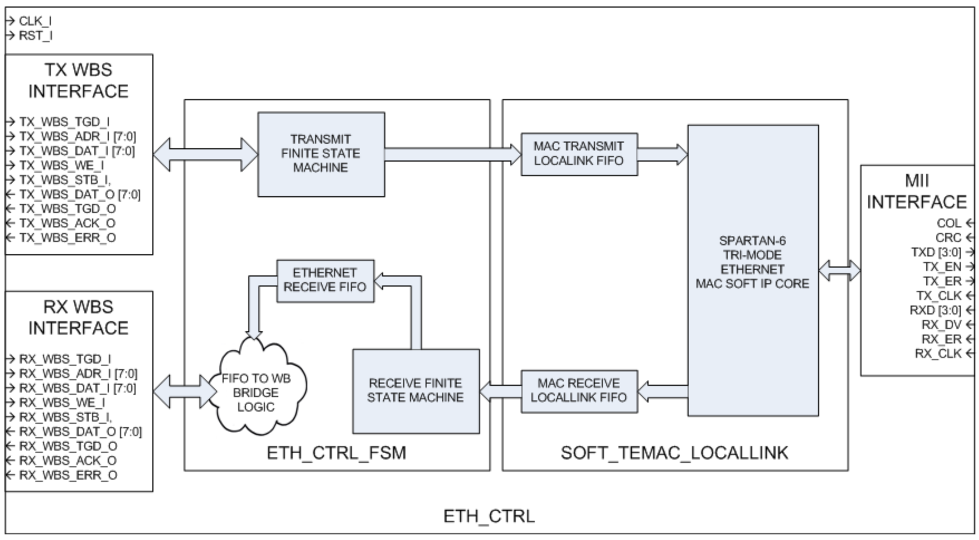


Figure 4.27: Ethernet Controller Module Functional Block Diagram

What is developed for this thesis work in Ethernet Controller Module is actually the ETH_CTRL_FSM unit. The operation of this unit is controlled by two internal finite state machines, which are namely Receive and Transmit FSMs. To understand the operation of the unit, the operation of these FSMs must be explained:

Receive Finite State Machine: This FSM, as seen in Figure 4.27, is responsible to read Ethernet frames from the MAC Receive *LocalLink FIFO* and to write them to a FIFO called

Ethernet Receive FIFO “after processing“ to be ultimately delivered to the upper layer, which is the WishBone Master module.

The packets read from the *LocalLink* FIFO are raw Ethernet frames, which are sourced from the control computer. Note that the operating system on the computer may send Ethernet frames of many different types. So, there exist many packets in the line except those targeting the *Acoustic Radar System*. Receive FSM reads all of these Ethernet frames and throws away the ones which are not targeting the system. As pointed out before, *Ethernet Controller* Module supports the User Datagram Protocol (UDP). User datagram protocol is a subset of Internet Protocol (IP). The IP packets are obviously carried on Ethernet frames, as seen from Figure 4.28. Receive FSM is responsible to check required fields of the frame and if all the conditions are satisfied the *Data* portion of the frame is written to the *Ethernet Receive FIFO* to be delivered to the upper layer.

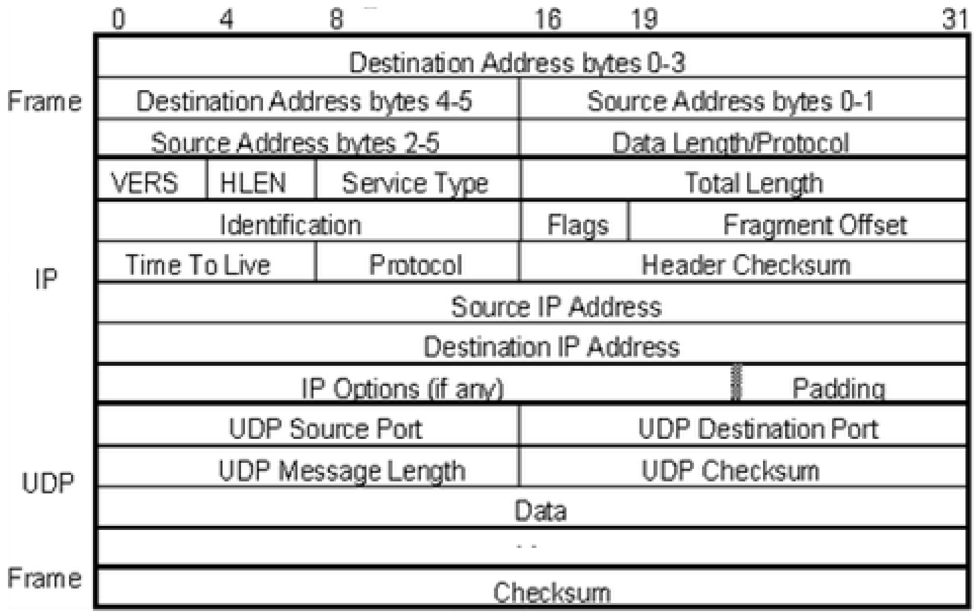


Figure 4.28: User Datagram Protocol Packet Format

Before leaving Receive FSM, one point to be explained is about how data is delivered to the upper layer WishBone Master. Note that, Receive FSM writes the user data to the *Ethernet Receive FIFO* and does not deal with delivery of it to the WishBone Master. A logic structure called “FIFO to WishBone Bridge Logic“ is implemented in ETH_CTRL_FSM Module to achieve the task of delivering the data present in *Ethernet Receive FIFO* to WishBone Master.

Transmit Finite State Machine: This FSM is responsible to form the UDP packets and write them to *Xilinx Transmit LocalLink FIFO* to be ultimately delivered to the Ethernet physical layer. It is triggered by a WishBone write access performed by the Master. There are internal WishBone accessible registers which must be configured by the Master prior to a transmission request. Those registers are nothing but the fields of the Ethernet, IP and UDP headers.

The fragmentation of the UDP data is also handled by the Transmit FSM. The upper layer WishBone Master does not deal with the length of the data to be transmitted, it just writes the whole data to the *Ethernet Controller* Module via performing WishBone accesses. If data length is less than the minimum UDP data length limit, Transmit FSM pads it to satisfy the requirement. Similarly, if the data length is more than the maximum UDP data length limit, Transmit FSM fragments the data and sends it in multiple UDP packets.

Port description of the Ethernet Controller module is given in Table A.9. The WishBone IOs are not explained individually, an explanation about the entire WishBone port is given instead. Individual explanations of the signals may be found in [23].

The port description of the module is given in Appendix A.9

4.2.10 Acoustic Radar System Controller Module

Acoustic Radar System Controller forms the main processing controller of the system. It employs an advanced finite state machine which controls the entire operation of the system. Being the main controller, the module has interfaces to many submodules (as seen from Figure 4.29). The application layer of the acoustic radar, which is the *System Configuration and Display Unit*, has a direct access to the internal registers of the system controller. These internal registers (given in Figure 4.29), which are to be explained shortly, are the parameters of the operation.

To explain the operation of the *Acoustic Radar System Controller* module, first of all, the descriptions of the system registers will be given. Then the internal finite state machine will be explained in detail.

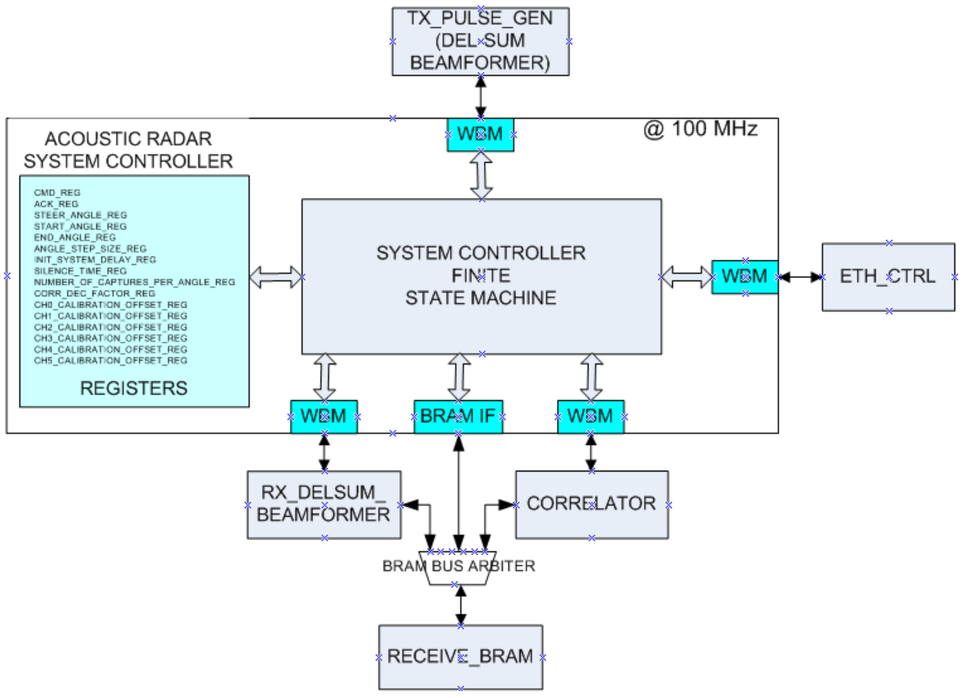


Figure 4.29: Acoustic Radar System Controller Functional Diagram

4.2.10.1 System Registers

Here, the descriptions of the internal system registers, which determine the operation flow and the parameters of the system, are given.

cmd_reg: The task is assigned to the module via accessing to this register. It is written by the acoustic radar user from the GUI, the possible values, i.e. tasks defined for the system are the following:

- system idle: When the system is powered-on or after reset button is pressed, the system starts in this mode. As the name implies, system is in idle mode when cmd_reg takes this value. When running in radar mode, the only way for the radar user application to stop the operation of the system is to configure cmd_reg as *system idle*.

- capture beamformer out: When this task is requested from the system controller, the output of the *Receive Delay and Sum Beamformer* module is sent to the user application to be ultimately plotted on a matlab figure.

- capture correlator out: When this task is requested from the system controller, the output of the *Correlator* module is sent to the user application to be ultimately plotted on a matlab figure.

- run radar: This is the *radar mode*. Environment is scanned and the radar information, which includes the correlation maximum index and value is sent to the user application. User application updates the radar screen accordingly.

ack_reg: When running in *radar mode*, a particular angle is listened and the radar information corresponding to that angle is sent to the user application. After that, the system *does not directly step* to the next angle, a feedback mechanism is employed in the system for robust operation. After necessary processing for the current angle is over, the user application sends an acknowledge packet to the system controller. The system controller steps to the next angle only upon this acknowledge. *ack_reg* holds the acknowledge information.

steer_angle_reg: When running in *capture beamformer out* or *capture correlator out* modes, system sends the beamforming/correlation output to the user application for a particular steer angle. This particular angle information is recorded in this register. In *radar mode*, steer angle information in this register is not used.

start_angle_reg: In *radar mode*, the system scans a region whose boundaries are defined by the user application. The starting angle information is recorded in this *start_angle_reg*

end_angle_reg: In *radar mode*, the system scans a region whose boundaries are defined by the user application. The end angle information is recorded in this *end_angle_reg*.

angle_step_size_reg: In *radar mode*, system scans the region with user defined steps. this register holds this step size information, as its name suggests.

init_system_delay_reg: Note that acoustic radar system relies on a process which includes transmitting a known pulse and then listening to the echo corresponding to that pulse. When *Transmit Pulse Generator* module is triggered to transmit the radar pulse, the waveform is not immediately emmitted to the air from the loudspeakers. Similarly, sound samples captured from the air by the microphones are not immediately written to channel FIFOs. So, the system has a characteristic transmit-receive *delay*. This delay should be taken into account by the system controller for proper operation. After the transmission of a pulse is triggered, channel

FIFOs must be hold in reset state during this initial system delay. Initial system delay is kept parametrical to observe the effect of different delay values. *init_system_delay_reg* holds the initial system delay information.

silence_time_reg: In the interval during which the radar pulse is being emitted from the loudspeakers, system should not listen to the air, i.e. channel FIFOs must be hold in reset until acoustic radar pulse is emitted to the air completely. This parameter is defined in order to observe the effect of varying silence time values on the decision process.

number_of_captures_per_angle_reg: In any mode, system conroller runs *Receive Delay and Sum Beamformer* module. Depending on the parameter *number of captures per angle* recorded in this register, beamforming is performed over one transmit-receive phase or multiple transmit-receive phases (Section 4.2.6).

corr_dec_factor_reg: Note that *correlator* (Section 4.2.7) module supports parametric decimation values. Decimation information is recorded in this register.

chx_calibration_reg: In phase calibration of six channels, channel waveforms are delayed by particular amounts (Section 4.2.6). The delay amount for a particular channel is determined by the user application via setting corresponding *ch_calibration_reg*.

4.2.10.2 Description of System Operation

As mentioned above, the entire operation of the acoustic radar system is controlled by a finite state machine which is employed in *Acoustic Radar System Controller* module. For a more systematical and easy-to-understand description of the state machine, it would be convenient to divide it into sub-phases. The state machine is formed by three fundamental parts, which are namely *observation*, *transmit-receive* and *data feed* phases, as shown in Figure 4.30.

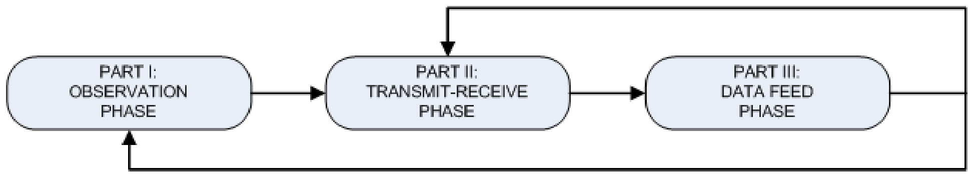


Figure 4.30: Acoustic Radar System Controller Finite State Machine Parts

Now, the operation in these three sub-phases will be explained in details.

PART I: Observation Phase

This is the phase in which finite state machine is continuously observing ethernet port for possible requests from the user application. The operational flow in this phase is given in Figure 4.31. In this phase, the user application can update any internal register (descriptions of registers are given in Section 4.2.10.1) of the system controller. Recall that possible requests from the user application is written to *cmd_reg*. When a valid request is written to *cmd_reg*, Part I is left and transmit-receive phase starts. The description of operation in individual states of *observation phase* may be explained as follows:

1. st00_reset: After power-up or when reset button is pushed, FSM enters this state. If reset button is released, this state is immediately left.

2. st01_init_clear_receive_bram After power-up, the default content of block RAMs is unknown unless otherwise specified. For proper operation of *Receive Delay-and-Sum Beamformer* module (and obviously the following modules which follows this module), the initial content of the *Receive BRAM* must be zero. So, as soon as reset is released, the first action that FSM takes is to clear entire *Receive BRAM*.

3. st02_check_cmd_reg: FSM checks *cmd_reg* for a valid request from the user application. If the task present in *cmd_reg* is *system idle*, FMS starts to observe ethernet port for a probable register access from the user application (*st03_read_from_eth*). If a valid task is present in *cmd_reg*, this state is left and *transmit-receive phase* starts. Valid tasks are either *capture beamformer out*, *capture correlator out* or *run radar*.

4. st03_read_from_eth: FSM observes ethernet port to receive possible register accesses from the user application. The packets received from the user application are UDP packets. A very simple protocol is designed for register accesses, in which the first byte is nothing but the address of the system register which is intended to be accessed and the following bytes are data to be written to that particular register.

PART II: Transmit-Receive Phase

This is the main phase of the state machine during which all signal processing of the acoustic radar system is employed. The operational flow in this phase is given in Figure 4.32. Transmit

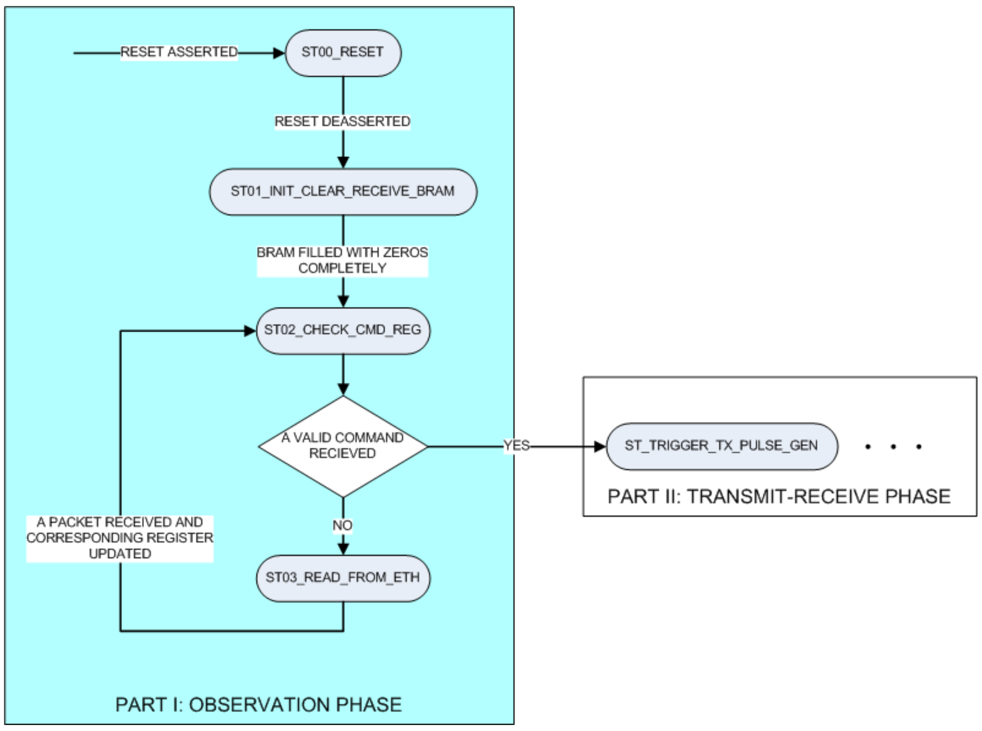

RESET ASSERTED
ST00_RESET
RESET DEASSERTED
ST01_INIT_CLEAR_RECEIVE_BRAM
BRAM FILLED WITH ZEROS COMPLETELY
ST02_CHECK_CMD_REG
A VALID COMMAND RECIEVED
YES
ST_TRIGGER_TX_PULSE_GEN
PART II: TRANSMIT-RECEIVE PHASE
NO
A PACKET RECEIVED AND CORRESPONDING REGISTER UPDATED
ST03_READ_FROM_ETH
PART I: OBSERVATION PHASE

Figure 4.31: Acoustic Radar State Machine Part I: Observation Phase

and receive beamforming and matched filtering (by correlation) all take place in this phase. Depending on the task requested by the user application, only *Receive Delay-and-Sum Beamformer* (for *capture beamformer out* task) or both *Receive Delay-and-Sum Beamformer* and *Correlator* modules (for *capture correlator out* and *run radar* tasks) are run by the system controller. The timing of operation which is obviously very critical for the system is also handled in this phase.

The description of operation in individual states of *transmit-receive phase* may be expained as follows:

1. st_trigger_tx_pulse_gen: As soon as *transmit-receive phase* is entered upon a valid request from the user application, the first action that FSM takes is to trigger *Transmit Pulse Generator* module via writing its *steer angle register* (see Section 4.2.1). The steer angle value with which *Transmit Pulse Generator* module is triggered may be either the value present in *steer_angle_reg* (Section 4.2.10.1) of system controller (for tasks *capture beamformer out* or *capture correlator out*) or the current radar angle which depends on *start_angle_reg*, *end_angle_reg* or *angle_step_size_reg* values (for

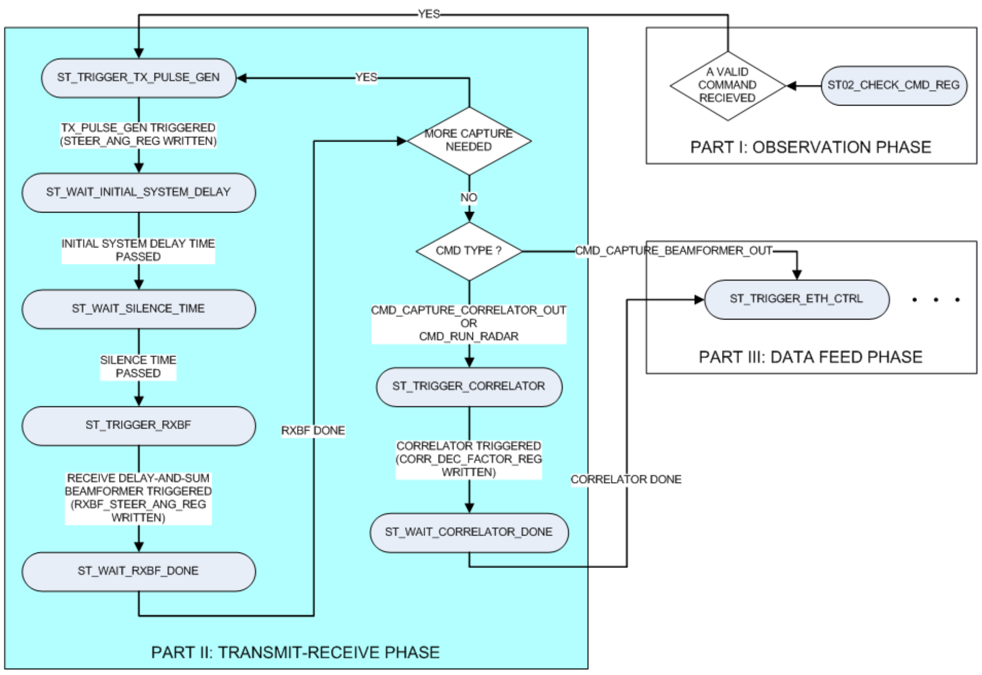

YES
ST_TRIGGER_TX_PULSE_GEN
YES
TX_PULSE_GEN TRIGGERED (STEER_ANG_REG WRITTEN)
MORE CAPTURE NEEDED
ST_WAIT_INITIAL_SYSTEM_DELAY
NO
INITIAL SYSTEM DELAY TIME PASSED
CMD TYPE ?
CMD_CAPTURE_BEAMFORMER_OUT
ST_WAIT_SILENCE_TIME
CMD_CAPTURE_CORRELATOR_OUT OR CMD_RUN_RADAR
SILENCE TIME PASSED
ST_TRIGGER_CORRELATOR
ST_TRIGGER_RXBF
RXBF DONE
RECEIVE DELAY-AND-SUM BEAMFORMER TRIGGERED (RXBF_STEER_ANG_REG WRITTEN)
CORRELATOR TRIGGERED (CORR_DEC_FACTOR_REG WRITTEN)
CORRELATOR DONE
ST_WAIT_CORRELATOR_DONE
ST_WAIT_RXBF_DONE
PART II: TRANSMIT-RECEIVE PHASE
A VALID COMMAND RECIEVED
ST02_CHECK_CMD_REG
PART I: OBSERVATION PHASE
ST_TRIGGER_ETH_CTRL
. . .
PART III: DATA FEED PHASE

Figure 4.32: Acoustic Radar State Machine Part II: Transmit-Receive Phase

run_radar task).

2. st_wait_initial_system_delay: What is meant by the phrase "initial system delay" is the time which passes between triggering of *Transmit Pulse Generator* module and arrival of the first corresponding sound samples to the channel FIFOs. In acoustical radar system, the front-end data capture blocks which are located before channel FIFOs (these blocks are *ADS8364 ADC Controller*, *DC Offset Remover* and *6-Channel FIR Low-Pass Filter*, see Figure 4.1) are continuously running. So, sound capture is always "on" and Channel FIFOs which are used to buffer data for further processing are always being written. Upon this explanation, an obvious question of how receive deactivation is implemented arises. It is actually very simple. *Acoustic Radar System Controller* module controls the *reset* of all channel FIFOs. It just holds all the FIFOs in *reset* state in which all write accesses to the FIFOs are ignored. So the deactivation of sound capture is effectively achieved in this way. Back to our discussion of initial system delay, after triggering *Transmit Pulse Generator*, in an ideal system, the acoustic radar pulse immediately would be emitted to air and captured by the receive blocks to the channel FIFOs. But, note that, in our implemented system structure, due to internal buffers in the structure and physical delays present in the transmit and receive cables,

a fixed delay is introduced. So, after pulse transmission is triggered, this initial delay must be waited by the FSM before activating sound capture, i.e. before deasserting resets of channel FIFOs. In acoustic radar experiments, this initial system delay is calculated as 1.6 milliseconds.

3. st_wait_silence_time: After initial system delay is waited by the FSM, sound capture still does not start immediately. Because the radar pulse emitted to air by transmit array must not be captured, the echoes corresponding to that transmitted pulse must be captured instead. So, FSM waits in this state until entire radar pulse is emitted to air. So, obviously, the wait time in this state is nothing but the duration of the acoustic radar pulse, which is 2 milliseconds.

4. st_trigger_rxbf: In this state, FSM activates sound capture via deasserting reset of Channel FIFOs and triggers *Receive Delay and Sum Beamformer* module via configuring its internal registers (first, six *chx_calibration_offset_reg*s are configured and then *steer_angle_reg* is updated with the current steer angle, see Section 4.2.6). During *transmit-receive phase*, this state may be visited multiple times depending on the *number of captures per angle* parameter. Multiple-visit to this state implies running *Receive Delay and Sum Beamformer* module multiple times, which would result in a cumulative beamformer output over multiple captures.

5. st_wait_rxbf_done: In this state, FSM waits *Receive Delay and Sum Beamformer* module to finish its operation. FSM observes *status register* of this module until *beamformer done* information is read. After beamforming is over, FSM checks whether more capture is needed or not, which is implied by the system parameter *number of captures per angle*. If more capture is needed, FSM goes back to the initial state of *transmit-receive phase* to trigger another pulse transmission. If no capture is needed, the next action FSM takes is to check the command type. If command type present in *cmd_reg* is *capture beamformer out*, there is no need to run *Correlator*, so FSM jumps to *data feed phase* directly. However, if command type is *capture correlator out* or *run radar*, *Correlator* module must be run.

6. st_trigger_correlator: In this state, *Correlator* module is triggered via writing its *decimation factor register* (see Section 4.2.7).

7. st_wait_correlator_done: In this state, FSM waits *Correlator* module to finish its oper-

ation. Similar to *Receive Delay and Sum Beamformer* module, *Correlator* signals that its operation is over by updating its *status register* with the value *correlator done*. After correlation is over, FSM direcly jumps to *data-feed phase*.

PART III: Data Feed Phase

This is the phase in which the information obtained for the current request is sent to the user application, which is *System Configuration and Display Unit*. In data-feed phase, depending on the current task, the following data may be sent to the user application.

- If current task is *capture beamformer out*, then the entire content of *Receive BRAM* which is nothing but the receive beamforming result is sent to the user application.
- If current task is *capture correlator out*, then the entire content of *Receive BRAM* which is nothing but the correlation result is sent to the user application.
- If current task is *run radar*, then the radar information which consists of the maximum correlation value and the index of this value is sent to the user application.

The flow of *data-feed phase* is given below in Figure 4.33. Due to space limitations, the states from which *data feed phase* is jumped could not be given. FSM jumps *data feed phase* only from *transmit-receive phase* (see Figure 4.32).

The description of operation in individual states of *data feed phase* may be expained as follows:

1. st_trigger_eth_ctrl: In this state, *Ethernet Controller* module is triggered to send a UDP packet to the user application. This module is triggered via writing *send packet instruction* to its *instruction register* (Section 4.2.9). After *Ethernet Controller* module is triggered, FSM jumps to the next state depending on the type of current task. If the current task is *capture beamformer out* or *capture beamformer out*, then FSM jumps to *st_bram_to_eth* to send entire *Receive BRAM* content to PC. Note that depending of these two tasks, the content of the *Receive BRAM* is either beamforming output of correlation output. If the current task is *run radar*, FSM jumps to *st_curr_radar_cycle_info_to_eth* state to send current radar information to the user application.

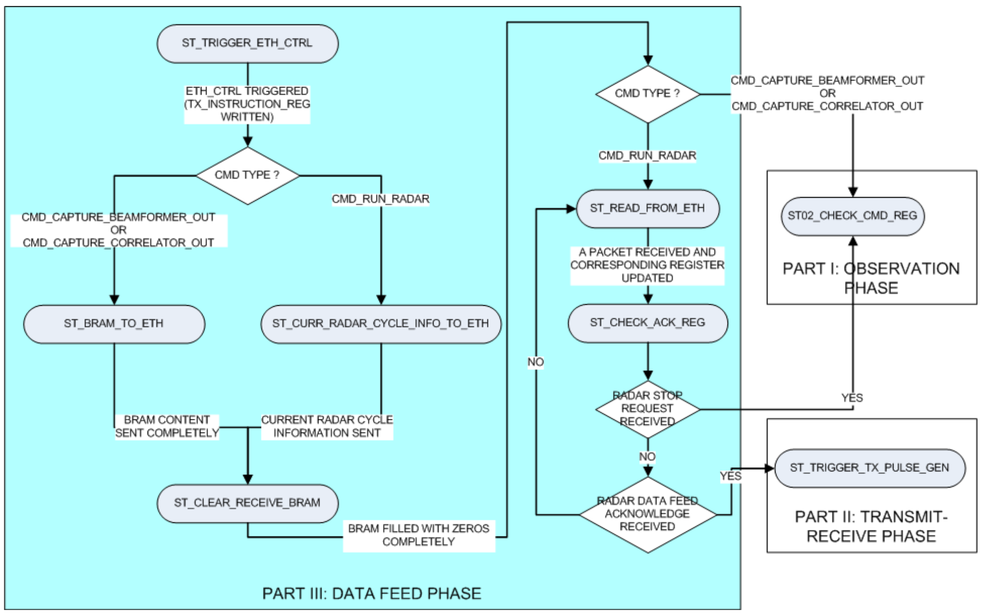

ST_TRIGGER_ETH_CTRL
ETH_CTRL TRIGGERED (TX_INSTRUCTION_REG WRITTEN)
CMD TYPE ?
CMD_CAPTURE_BEAMFORMER_OUT OR CMD_CAPTURE_CORRELATOR_OUT
CMD_RUN_RADAR
ST_BRAM_TO_ETH
ST_CURR_RADAR_CYCLE_INFO_TO_ETH
BRAM CONTENT SENT COMPLETELY
CURRENT RADAR CYCLE INFORMATION SENT
ST_CLEAR_RECEIVE_BRAM
BRAM FILLED WITH ZEROS COMPLETELY
PART III: DATA FEED PHASE
CMD TYPE ?
CMD_CAPTURE_BEAMFORMER_OUT OR CMD_CAPTURE_CORRELATOR_OUT
CMD_RUN_RADAR
ST_READ_FROM_ETH
A PACKET RECEIVED AND CORRESPONDING REGISTER UPDATED
ST_CHECK_ACK_REG
NO
RADAR STOP REQUEST RECEIVED
YES
NO
RADAR DATA FEED ACKNOWLEDGE RECEIVED
YES
ST02_CHECK_CMD_REG
PART I: OBSERVATION PHASE
ST_TRIGGER_TX_PULSE_GEN
PART II: TRANSMIT-RECEIVE PHASE

Figure 4.33: Acoustic Radar State Machine Part III: Data Feed Phase

2. st_bram_to_eth In this state, entire *Receive BRAM* content is written to *Ethernet Controller* module to be ultimately sent to the user application.

3. st_curr_radar_cycle_info_to_eth: In this state, current radar information, which consists of

   - the **index** of the correlation maximum (4 bytes) and
   - the **value** of the correlation maximum

   is sent to the user application.

4. st_clear_receive_bram: No matter what type of task is processed, *Receive BRAM* must be cleared at the end of a *data feed phase* in order to make it ready to be used for future accesses. Clearing *Receive BRAM* means filling it with zeros entirely. After *Receive BRAM* is cleared, FSM checks the current task. If the current task is *capture beamformer out* or *capture correlator out*, then FSM goes back to the initial phase, which is *observation phase*. Because for these tasks, no more *transmit-receive phase* is necessary. But, if the current task is *run radar*, then another *transmit-receive phase* will be run after a feedback packet is received from the user application. If the current task is *run radar*, FSM jumps to the initial phase, which is *observation phase* only if

the user application cancels the operation.

5. st_read_from_eth: The operation in this state is exactly same as the one in *observation phase*. This state is employed in *data feed phase* in order to provide the user application the ability to update any internal system register. So, while the system is operating in radar mode, the parameters may be changed real-time without intervening the operation.

6. st_check_ack_reg : In radar mode, after the information corresponding to a particular angle is sent to the user application to be ultimately used in updating the radar screen, FSM does not immediately jumps to the next angle. It first checks the content of *cmd_reg*. If *sytem idle* task is present in this register, FSM jumps to the initial phase, which is the *observation* phase. This is the case that radar mode is interrupted. If this is not the case, i.e. if the user application does not inturrupt radar operation via writing *system idle* to *cmd_reg*, the next action that FSM takes is to check *ack_reg*. If feedback information is present in this register, which implies that the user application is processed the current radar information and the next angle can be jumped, FSM jumps back to *transmit-receive phase* in order to perform another radar cycle for the next angle. If acknowledge is not received yet, FSM jumps back to st_read_from_eth in order to receive possible register accesses from the user application.

In this part, Receive BRAM is explained. Note from Figure 4.1 that the *Receive BRAM* (which is referred multiple times in this section) is used by three modules, namely, *Receive Delay and Sum Beamformer*, *Correlator* and *Acoustic Radar System Controller*. The reason why such a shared structure is employed in the system is to minimize FPGA memory resource utilization. This is a BRAM with such a high capacity that entire capture is recorded to which (capture length = 14112 samples, each of which is 16 bits). Note that there is only one physical port to access BRAM, so, this port somehow must be devoted to one of three candidates. This is achieved by the system with an arbiter module, which is called *BRAM Bus Arbiter*. This module has a *port select* input. Depending on the value of this *port select* input, *BRAM Bus Arbiter* switches the physical port of the *Receive BRAM* to a particular master. The *port select* input is -as expected- controlled by the *Acoustic Radar System Controller*, being the intelligence in the system. So, *Acoustic Radar System Controller*, being aware of the module which is to use *Receive BRAM*, sets the *port select* of the arbiter such that the particular module will have a direct access to BRAM.

CHAPTER 5

EXPERIMENTAL RESULTS

During the development of the Acoustic Radar System, many tests on hardware have been performed. This tests are actually the integral part of the system design and implementation process. The very initial hardware tests are performed to validate the transmit part of the system, which are explained in detail in Section 5.1. At that time, the receive part of the system was missing, actually. Then, having validated the transmitter part, the receiver units are added to the overall system one by one. So, an incremental system development method is followed. Receiver hardware tests, which are explained in detail in Section 5.2, are mainly categorized as two parts, namely, *ADC Capture over UDP* (in which 6 Channel ADC data is captured to computer for analyzing) and *Acoustic Radar System Evaluation* (in which the ultimate system validation tests are performed).

5.1 PC Controlled Transmit Beamformer

The purpose of this implementation is to verify the transmitter part of the Acoustic Radar System. This implementation provides user the ability of controlling the transmit beamforming process by sending commands from MATLAB. The software application running on MATLAB communicates with the FPGA via RS232 Serial Communication.

This may be considered as a partial implementation of the entire system. Some modifications have been made on the original system. This modification can be seen in Figure 5.1). In the *Acoustic Radar Top Module*, the modules *PCM1602 DAC Controller* and *Transmit Pulse Generator* are preserved in their original forms, while the receive modules are removed from the system. Instead of the ultimate *Radar System Controller* Module, a simpler and special module has been designed and implemented, which is called *"UART_TO_PULSE_GEN_FSM"*. Another difference from the original system is that RS232 Serial Protocol is used instead of UDP for communication with the Computer. For this purpose, a unit called UART_CTRL is designed, implemented and employed in the system. The reason why UART communication is used instead of Ethernet in this implementation is that the development of the Ethernet Controller module which implements the UDP (User Datagram Protocol) had not been completed at that time.

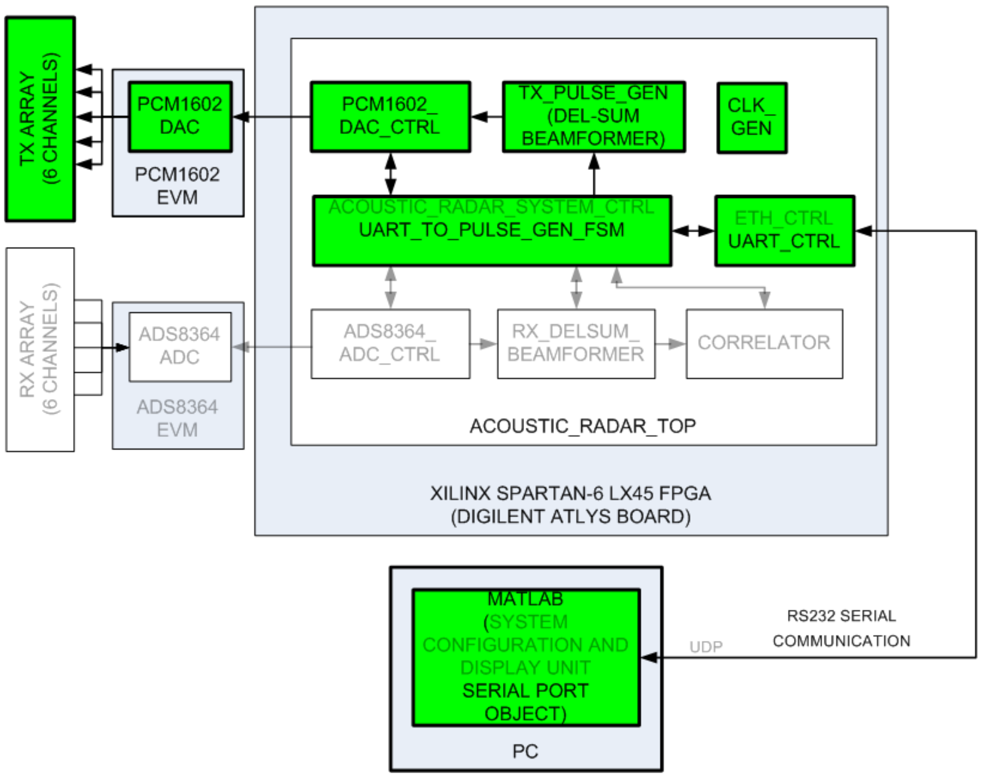


Figure 5.1: PC Controlled Transmit Beamformer Functional Block Diagram

5.1.1 5.1.1 Description of the System Parameters

Before describing the system operation, it would be better to explain the real time controllable features of the system. There are three parameters that may be set by the user:

1. *Pulse Transmission Type*: There are two kinds of pulse generation pattern in the implemented beamformer: *single* or *infinite*. In *single mode*, system generates only a single

pulse with the desired steer angle and then stops. But, when it is operated in *infinite mode*, the system generates periodic pulses continuously (until it is interrupted by the user by a single mode command) with a user defined period (which is called *Pulse Repetition Period* parameter). While operating in *infinite mode*, the user can change the *Steer Angle* of the beamformer in real-time without interrupting the operation of the system.

2. *Steer Angle*: This is the look direction of the transmit beamformer. *Steer Angle* parameter may take values [10, 170] degrees with a resolution of 10 degrees. In *Infinite Mode*, as pointed out above, the *Steer Angle* parameter may be changed by the user with a command without intervening the operation of the system.

3. *Pulse Repetition Period*: As pointed out above, while operating in *Infinite Mode*, the system generates periodic pulses. The period of these pulses are called Pulse Repetition Period and set by the user in real-time via MATLAB commands. This parameter may take values 0 to 27 seconds in a 200 ns resolution.

5.1.2 Description of System Operation

Now, after explaining the real-time operation parameters of the system, the functional descriptions of the special modules designed for this experiment may be given as follows.

5.1.2.1 UART_TO_PULSE_GEN_FSM MODULE

This module is the main controller of the PC Controlled Transmit Beamformer Implementation, which may be thought as an ancestor of the ultimate *Acoustic Radar System Controller* Module.

The operation of the module is controlled by an internal finite state machine whose flow diagram is given in Figure 5.2. As soon as the system reset is released (it is a hard reset connected to an on-board switch), the FSM starts to observe *UART_CTRL request register* (*observe_uart_port* state, via performing WishBone [23] read accesses, WishBone is explained in 4.2.1) for a probable data reception from the serial port. When a data reception is observed from the serial port, FSM performs a read access (*read_uart_data* state) to the *rx data* *register* of the UART_CTRL. The 64 bit received data read from rx data register contains all information about the system operation parameters, namely, *Pulse Pattern*, *Steer Angle* and *Pulse Repetition Pattern*, whose are explained in 5.1.1. Then, depending on the request type, FSM changes state to trigger *Transmit Pulse Generator* Module 4.2.1. If the request implies *Single Pattern*, then FSM triggers pulse generator module with the intended *Steer Angle* (*single_trigger_to_pulse_gen* state) then starts observing the serial port again. Otherwise, i.e. if the request implies *Infinite Pattern*, FSM starts to trigger *Transmit Pulse Generator* infinitely with a period of *Pulse Repetition Period* (*run_pulse_gen_infinitely* state) and with an intended look direction, i.e. *Steer Angle*. The *Pulse Repetition Period* and *Steer Angle* parameters can be changed in real-time by the user. FSM leaves this infinite pulse triggering state only if a *Single Pattern* request is received from the UART_CTRL. So, it is obvious that even in the infinite triggering state, the FSM still observes the UART_CTRL WishBone port for a probable request from the user. Yet, otherwise, the fact that the *Steer Angle* and *Pulse Repetition Period* can be changed in real-time would be impossible.

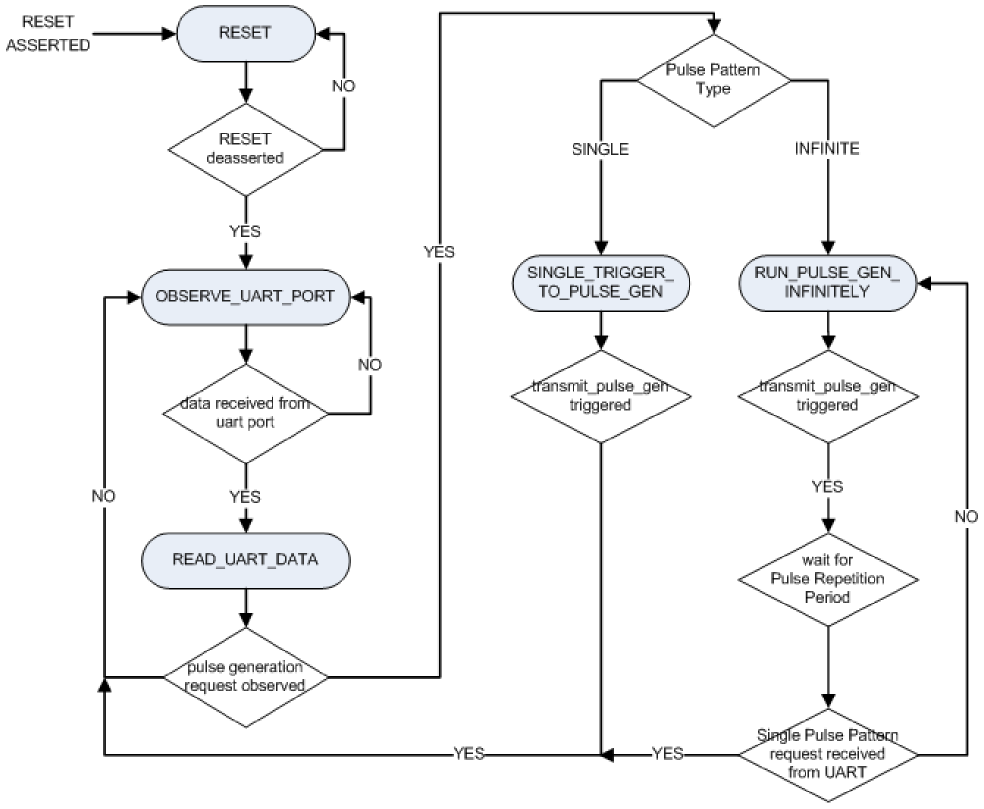

RESET ASSERTED
RESET
NO
RESET deasserted
YES
OBSERVE_UART_PORT
NO
data received from uart port
YES
READ_UART_DATA
pulse generation request observed
NO
YES
Pulse Pattern Type
SINGLE
INFINITE
SINGLE_TRIGGER_TO_PULSE_GEN
RUN_PULSE_GEN_INFINITELY
transmit_pulse_gen triggered
transmit_pulse_gen triggered
YES
wait for Pulse Repetition Period
NO
Single Pulse Pattern request received from UART
YES
YES

Figure 5.2: PC Controlled Transmit Beamformer Finite State Machine Flow Diagram

5.1.2.2 UART_CTRL

This module is designed and implemented to achieve the Serial Communication between FPGA and the PC. In addition to handling the low level Serial Communications protocol signalling, the module also implements a higher level which provides a simple hand-shake protocol for the PC-FPGA communication.

In this module, to handle the physical layer signalling issues, an open source unit called “UART CORE“ has been used. For upper layer communication with the WishBone Master, two state machines have been designed and implemented, which are transmit and receive state machines (Figure 5.3).

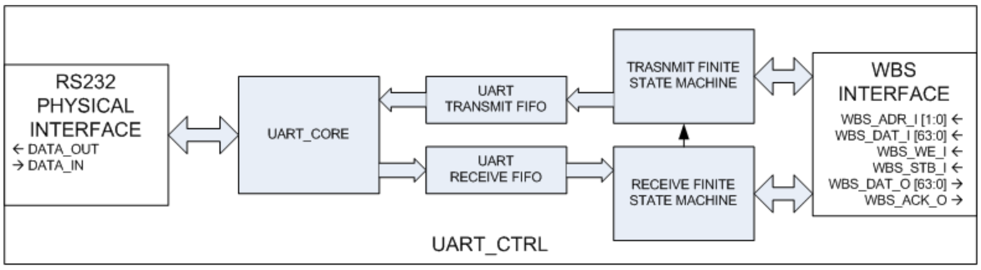


Figure 5.3: UART Controller Module Functional Block Diagram

Receive FSM is responsible to deliver packets received by the *UART_CORE* IP to the WishBone Master. The operation is as follows: It observes the receive FIFO. When the empty condition of the FIFO is over, i.e. there is data in the FIFO, It checks an internal register called *status register* which holds the information whether the previous received data is read by the WishBone Master or not. If the previous read data from the FIFO has not been read by the WishBone Master, the Receive FSM simply waits and does not read the Receive FIFO for new data; otherwise the previous data would be overwritten and lost. If the previous data has been read by the WishBone Master, Receive FSM reads the FIFO and updates the internal Receive Data Register with this new data. Moreover, data read from the FIFO is not a raw one, the designed handshake protocol requires a 4 bytes header, Receive FSM checks this header and throws away the packet if it is not as expected.

Transmit FSM is responsible to deliver the Acknowledge messages and WishBone Master data to the UART_CORE IP to be ultimately transmitted onto the Physical Line. Transmit

FSM is not completely isolated from the Receive FSM, it is triggered to send an Acknowledge Message to the computer when a received data is supplied to the Master successfully, i.e. when WishBone Master reads the *Receive Data Register* successfully. Acknowledge message is a part of the designed simple handshake protocol. It informs the Computer that its transmitted packet is delivered to the FPGA upper layer WishBone Master successfully.

Note that UART Controller Module has only one WishBone interface, so simultaneous transmit and receive is not possible.

5.1.3 Results

The PC Controlled Transmit Beamformer is tested on the Acoustic Radar Hardware described in 4.1. A view of the setup is given in Figure 5.4. System is controlled via sending commands from MATLAB software, which contains information of *pulse pattern*, *steer angle* and *pulse repetition period* parameters.

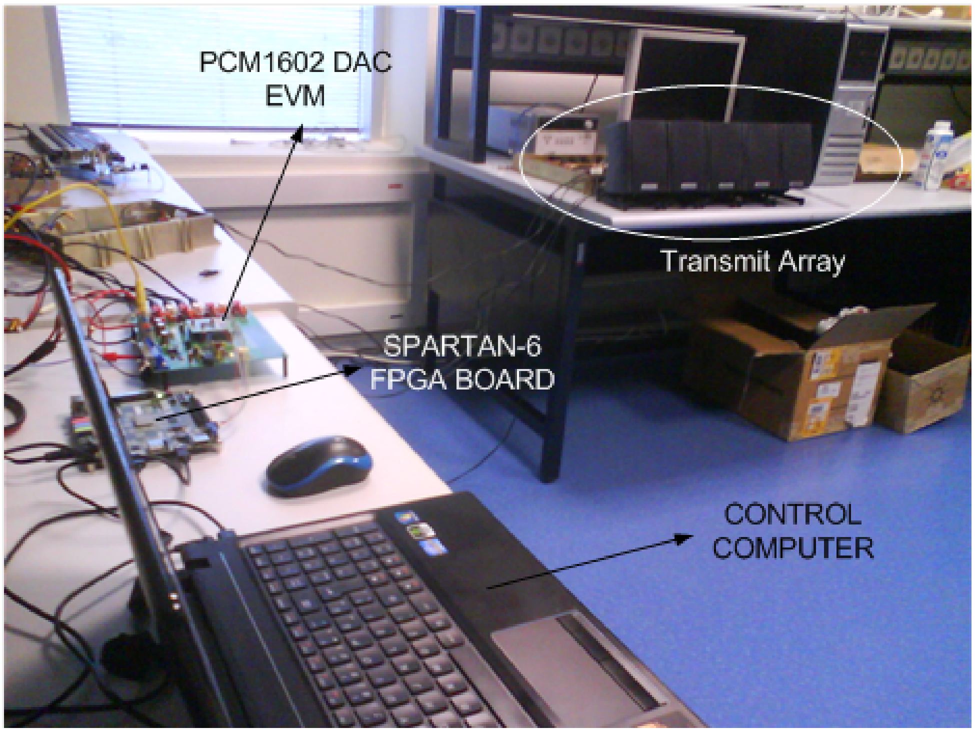


Figure 5.4: PC Controlled Transmit Beamformer Test Setup

For the validation of the system, XILINX ChipScope Tool [37] and an oscilloscope are used.

ChipScope is a tool which provides the user the ability of debugging the implementation running on FPGA real-time. In order to debug an FPGA design with ChipScope software, the designer adds some special Cores to the design before implementation. So, an arbitrary FPGA configuration cannot be debugged using ChipScope Analyser software, the implementation which runs on FPGA must be generated with those special ChipScope IP Cores just mentioned.

Using ChipScope Analyser software, the output of the Transmit Pulse Generator Module is observed. In order to validate the output of the module, the only condition that must be checked is the delay between the channels.

In Figure 5.5, the observation for 80° steer angle is given. The delay between successive channels in order to achieve 80° steer angle is calculated as 61.29 μsec, which approximately corresponds to 3 samples. This is exactly the observed case in Figure 5.5. Note that for 80° steer angle case,

the leading channel is channel 5, which is as expected. (for steer angle values of [0, 90], the leading channel is channel 5, but for steer angle values of [90, 180] degrees, the leading channel is channel 0)

Similarly, for the 170° case, the delay required between successive channels is 374 μsec, which approximately corresponds to 18 samples. This is exactly the observed case in Figure 5.6. The negative sign in the delay calculation implies that the leading channel is channel 0 (Note that for 80° case, the leading channel was channel 5).

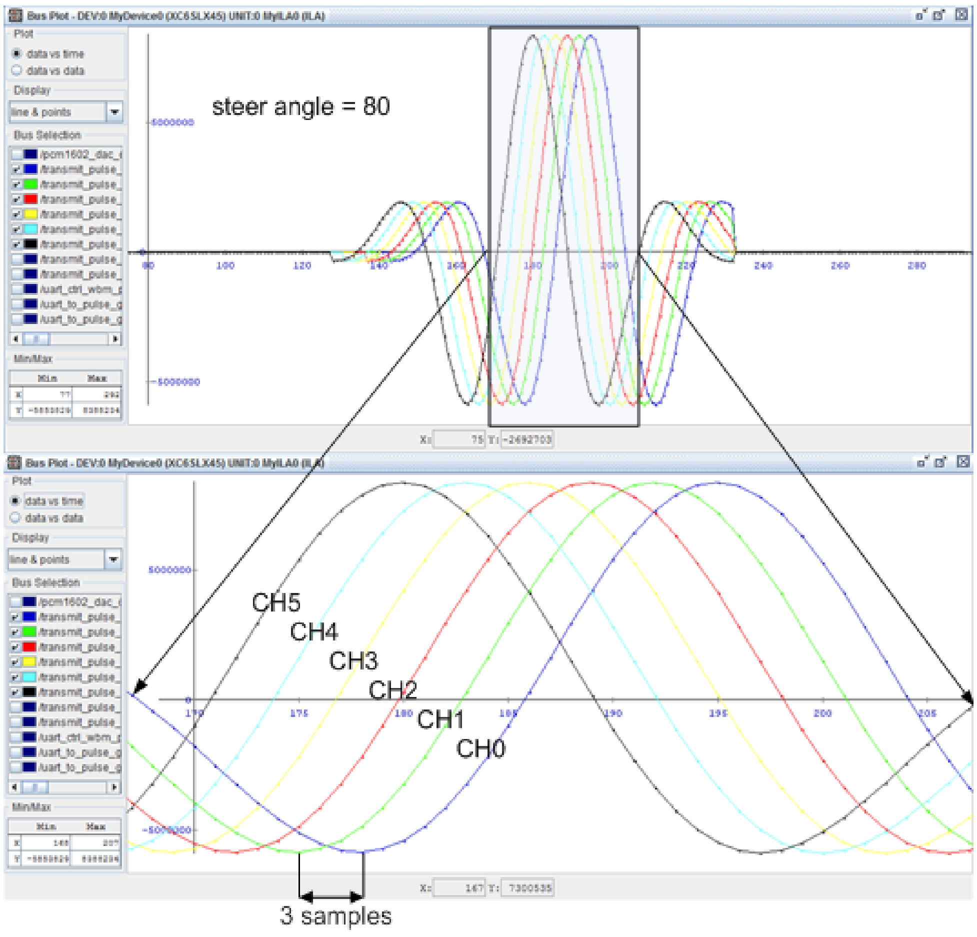


Figure 5.5: *Transmit Pulse Generator* output, *steer angle* is 80°

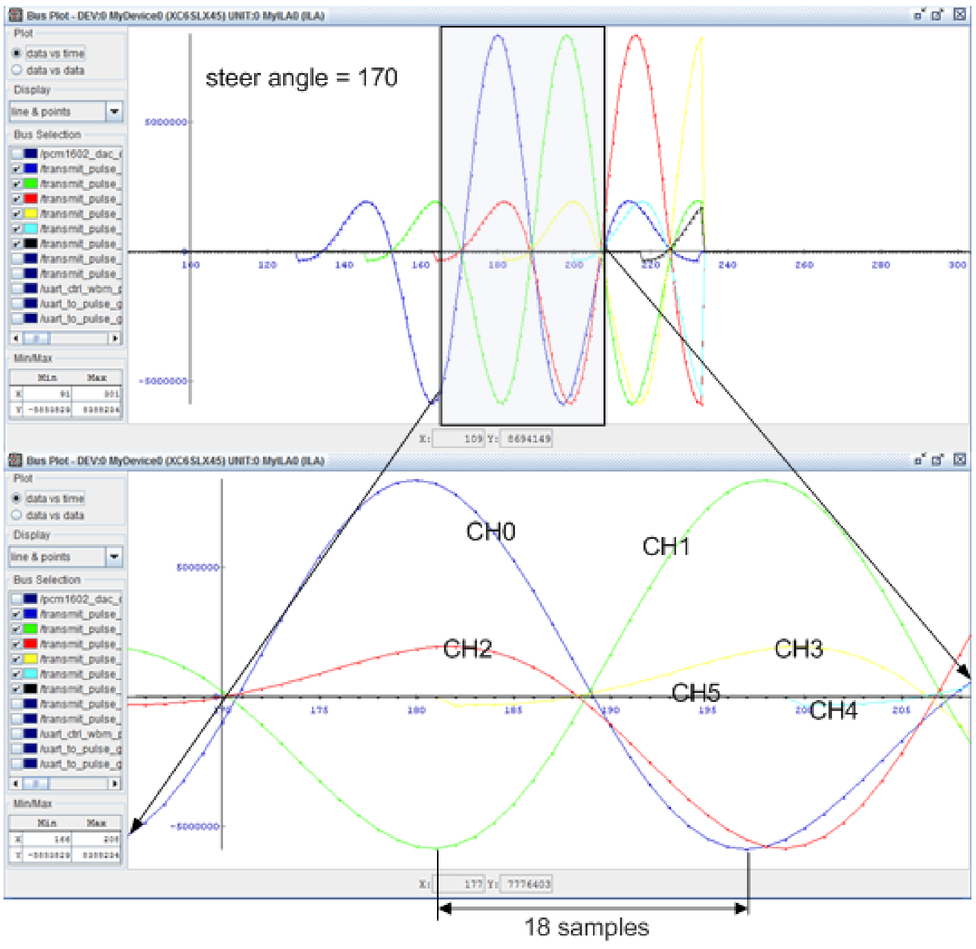


Figure 5.6: *Transmit Pulse Generator* output, *steer angle* is 170°

The *infinite mode* is also validated using ChipScope Analyzer software. It can easily be observed from Figure 5.7 that *Transmit Pulse Generator* module generates periodic pulses which is consistent with the required *pulse repetition period* parameter. In Figure 5.7, for 170 ° steer angle case, the *Transmit Pulse Generator* output is given for three different *pulse repetition period* parameters.

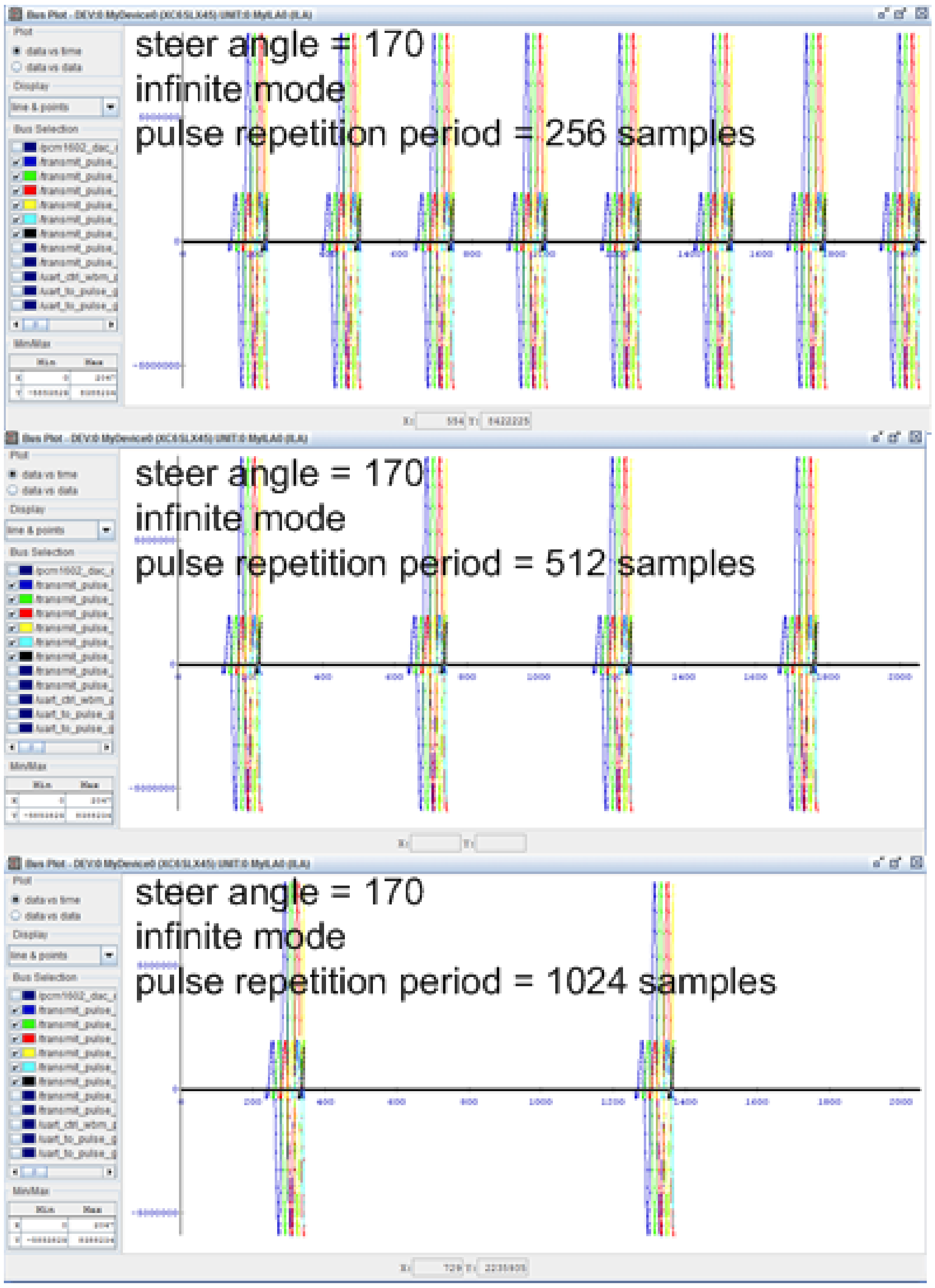

steer angle = 170
infinite mode
pulse repetition period = 256 samples
steer angle = 170
infinite mode
pulse repetition period = 512 samples
steer angle = 170
infinite mode
pulse repetition period = 1024 samples

Figure 5.7: Transmit Pulse Generator output in *infinite mode* for three different *pulse repetition parameter* values

5.2 Acoustic Radar Experiments

In this section section, hardware tests, which form the integral part of acoustic radar system development, are explained and the corresponding results are given. The ultimate system given in Figure 4.1, obviously has not been directly implemented on hardware, an incremental implementation procedure is followed. First of all, a partial system with some missing receiver blocks is tested, which is explained in Section 5.2.1 . In this hardware tests, 6-channel ADC capture (together with DC removal and low-pass filtering) is validated, no beamforming and cross-correlation blocks are employed. After validating adc capture, the ultimate system is implemented and tested on hardware, which is explained in 5.2.2.

5.2.1 Experiment 1: ADC Capture over UDP

Before designing the ultimate acoustic radar system, hardware tests are conducted on the partial structure given in Figure 5.8. Actually, this partial system is itself developed in an incremental manner. First release of the ADC capture system had just included raw ADC data capture, i.e. neither dc offset removal nor low-pass filtering had been employed. Then, after they were designed and validated by ModelSim [38] simulations, the *DC Offset Remover* and *6-Channel FIR Low-Pass Filter* modules were added. The differences between the partial implementation and the ultimate system can be seen in Figures 5.8 and 4.1. This structure does not implement beamforming and correlation. Also, instead of the ultimate acoustic radar system controller module, a specific controller called *ADC_Capture_Over_UDP* is designed and implemented.

There are there parameters (which can be set by the user application running on MATLAB software) in this hardware test setup, which are

1. Steer Angle
2. Pulse Repetition Frequency
3. Capture Length

When *start capture* command is sent to the system from the user application, first of all, *Transmit Pulse Generator* module is triggered to transmit a pulse to the direction implied

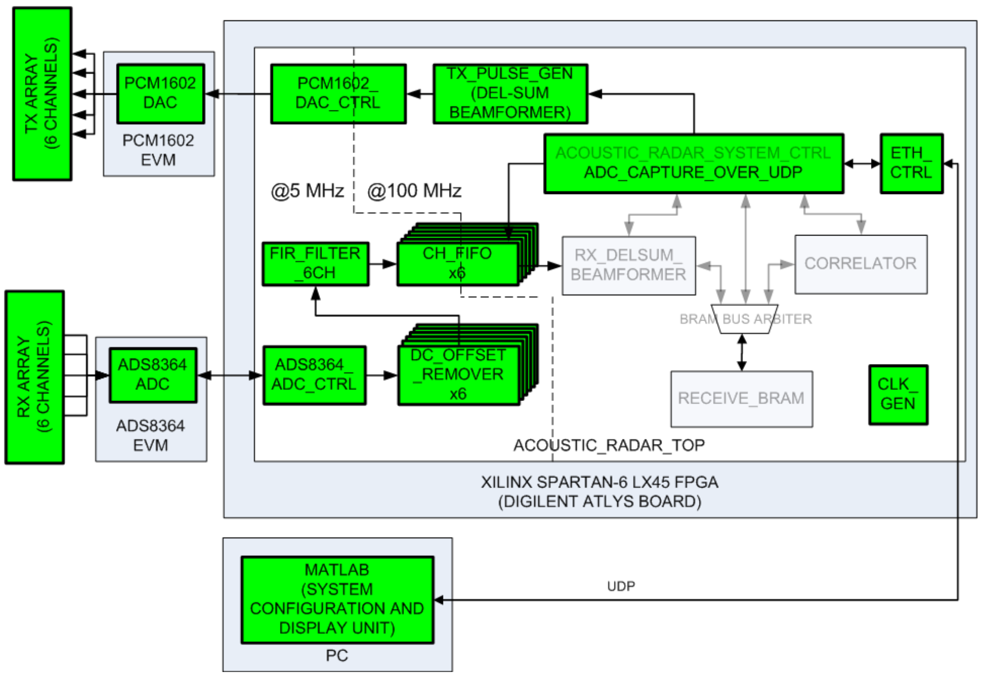


Figure 5.8: ADC Capture over UDP Functional Block Diagram

by the *steer angle* parameter. As soon as *Transmit Pulse Generator* module is triggered, the reset of all channel FIFOs are released and the content of these FIFOs are sent to the user application over UDP for a duration implied by the *Capture Length* parameter.

Figure 5.9 is obtained in a setup on which the target is placed 3 meters away from the transmit and receive arrays with an angle of 90°. This corresponds to the array broadside. When no DC Offset Removal and low-pass filtering is employed the above capture is obtained. It has a very high DC offset value and it is not clean because of high frequency components and environmental noise. The improvement introduced by the modules *DC Offset Remover* and *6-Channel FIR Low-Pass Filter* is obvious in Figure 5.9.

In the following parts, only the low-pass filtered results are presented.

When ADC Capture over UDP system is run with a pulse repetition frequency covering 10 meters (which is 58.8 msec) and a capture length equal to one pulse repetition period, the following result in Figure 5.10 is obtained. In Figure 5.10, all channel waveforms are plotted. The first pulse with a higher magnitude is the pulse transmitted and the following one is the echoed one from the target which is 3 meters away from the system. Note that the time

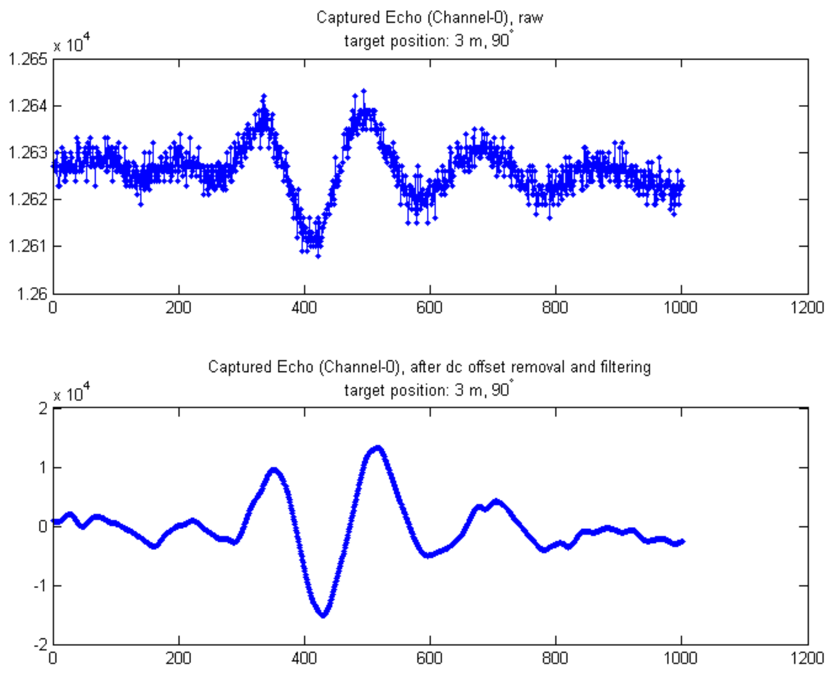


Figure 5.9: Channel-0 Captured Echo Comparison, Raw vs DC Offset Removed, Low-Pass Filtered Waveform

between the transmitted and received pulses is 17 milliseconds. The distance that sound travels in 17 milliseconds = 17 msec $\times$ 346.13 m/sec = 5.8842 meters (speed of sound at 25° temperature = 346.13 m/sec). So, the target is at half the distance travelled by the sound, namely, 2.94 meters, which is almost 3 meters, as expected. Note also from Figure 5.10 that the length of the echo pulse is 2 milliseconds, which is the acoustic radar pulse length.

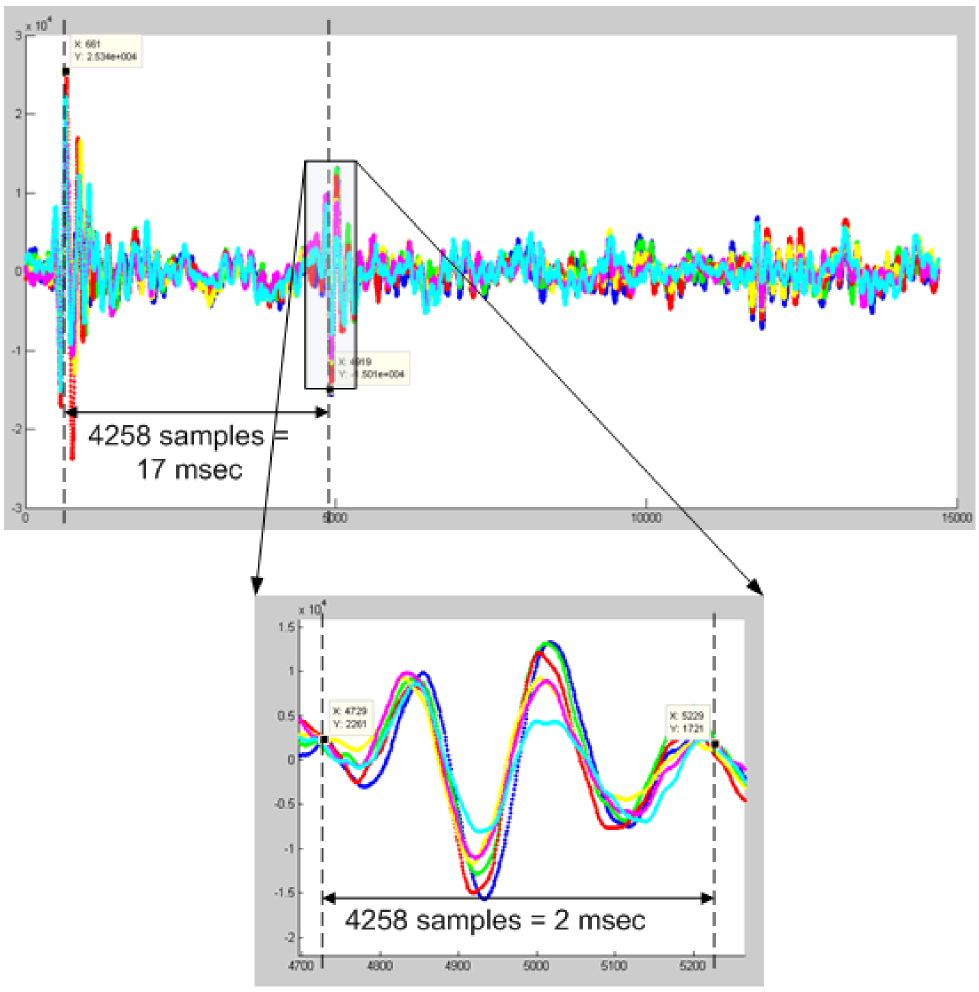


Figure 5.10: 6-Channel Capture, duration: 58.8 msec (1 PRF), target position: 3 m, 90°

5.2.2 Experiment 2: Acoustic Radar System Evaluation

After validation of the ADC capture from 6 channels using the testbed above, whose structure is given in Figure 5.8, as the last step of this thesis work, the ultimate acoustic radar system is implemented on hardware, whose structure is given in Figure 4.1.

The results given in this section are obtained in hardware tests in which the target is placed in two different positions (as seen from Figures 5.11 and 5.12), which are

1. distance: 3.5 m, direction: 90°
2. distance: 2 m, direction: 60°

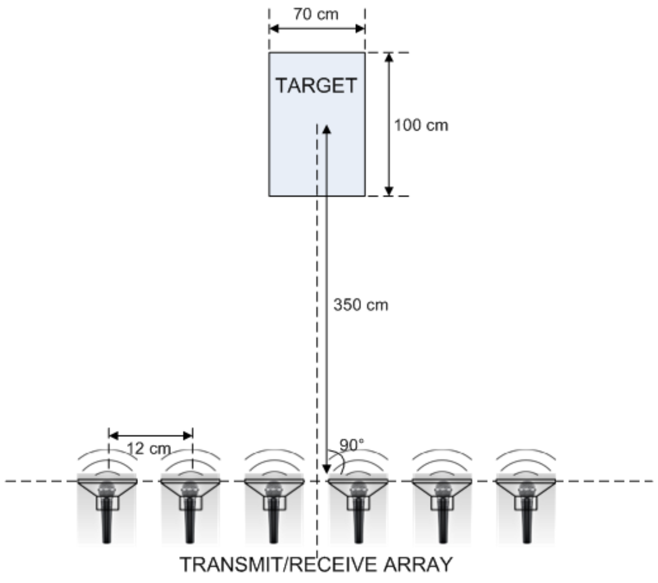


Figure 5.11: Target placement for Scenario 1

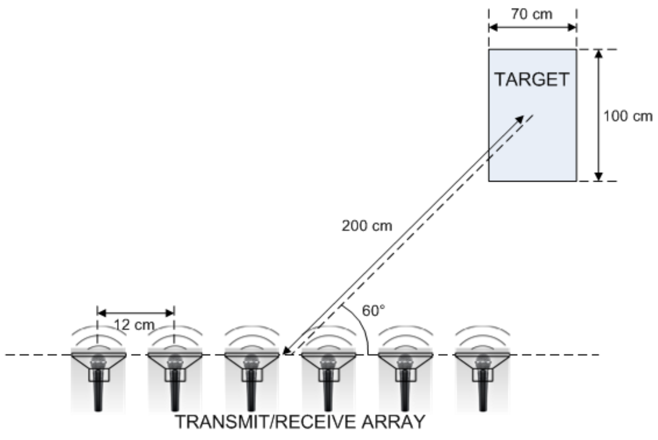


Figure 5.12: Target placement for Scenario 2

For these cases, the entire radar cycle is observed and the corresponding signal captures are given in the following two parts. The observed waveforms are, namely, transmit array output, echo received by the receive array, the beamformer output (with *number of captures per angle* parameter varied, refer *number_of_captures_per_angle_reg* explanation in Section 4.2.10.1) and the correlator output (with *correlator decimation factor* parameter varied, refer *corr_dec_factor_reg* explanation in Section 4.2.10.1). Finally, the resulting radar screen shot corresponding to the current test angle is given.

Scenario 1

Transmitted Pulse Observation: In the current scenario, first, *Transmit Pulse Generator* module is triggered for 90 ° steer angle. The sound waveforms which are emitted to air from the transmit array are observed in the receive array's point of view. In the following two figures, namely, Figure 5.13 and Figure 5.14, the transmitted waveforms observed from the receive array (after dc offet removal and low-pass filtering) for two neighbouring channels and for all channels are given, respectively.

Note that for 90° steer angle, the delay between successive channels is zero.

In other words, there is no phase difference between channels for 90° steer angle, as expected. It is exacty the case in figures 5.13 and 5.14. Actually, what is observed in these figures is not the the exact waveforms emitted to air from the transmit array. Because, as mentioned above, they are waveforms captured by the receive array. Note that transmit and receive arrays are very close to each other, so the transmitted waveform cannot be captured by the receive array perfectly because of the near-field effect. But, despite being distorted slightly, it gives a satisfying idea about the transmitted waveforms (Actually, the transmitted waveforms are already validated by the hardware tests performed in Section 5.1).

Note from figures 5.13 and 5.14 that the transmitted waveforms are not immediately captured by the receiver, there is a fixed offset (it is 1.6 msec) between triggering of *Transmit Pulse Generator* and presence of corresponding radar pulse in receive array. This is called "initial system delay" (refer explanation of *st_wait_initial_system_delay state* of *Acoustic Radar System Controller* in 4.2.10.2).

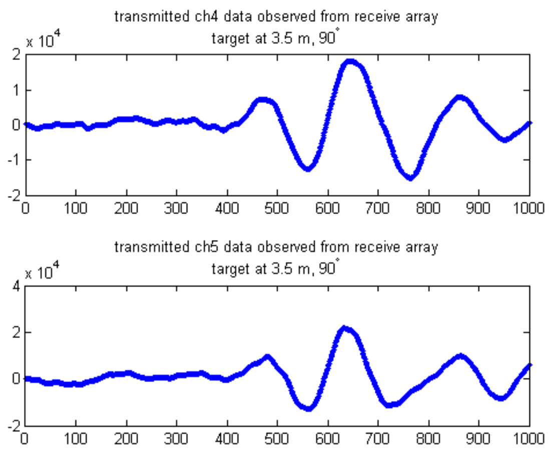


Figure 5.13: Transmitted CH4 and CH5 waveforms from the receive array's point of view, no phase difference observed, target position: 3.5 m, 90°

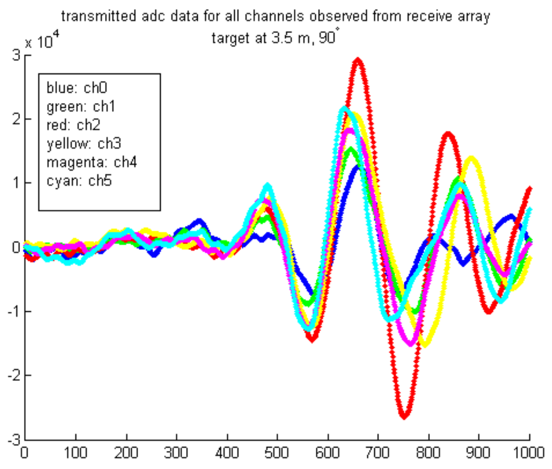


Figure 5.14: Transmitted waveforms for all channels from the receive array's point of view, no phase difference observed, target position: 3.5 m, 90°

Echo Pulse Observation: The transmitted waveforms propagate through air, hit to the target and are reflected back to the system. The received echo observed in two successive channels and in all channels are given below in figures 5.15 and 5.16, respectively. Note that, there is no phase difference in channel waveforms, which is expected. There is also mismatch between channels in the receive array. The signal magnitudes for different channels are different, which is actually not a problem for the system functionality. Far-field effect may also be observed if the figures 5.14 and 5.16 are compared. Note that the received echo signal is less distorted and closer to the acoustic radar pulse in shape than the transmitted pulse in the receiver array's point of view.

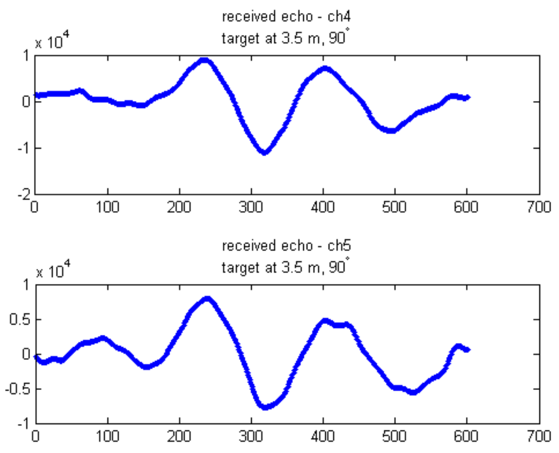


Figure 5.15: Received echo from CH4 and CH5, no phase difference observed, target position: 3.5 m, 90°

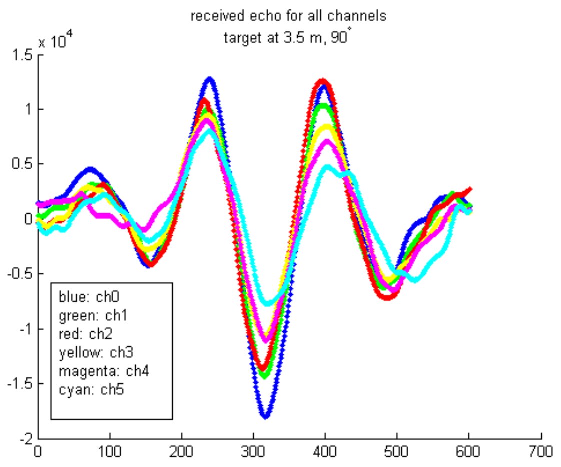


Figure 5.16: Received echo from all channels, no phase difference observed, target position: 3.5 m, 90°

The transmitted and received pulses are given above in separate plots from the receive array's point of view. The entire transmit-receive cycle is given below in Figure 5.17. Note that the time of flight of the transmitted radar pulse in air is 19.6 milliseconds as given in the figure. It corresponds to a distance of 19.6 msec×346.13 m/sec = 6.7883 m (speed of sound at 25° temperature = 346.13 m/sec). So, it means that the target is at 6.7883/2 = 3.3941 ~ 3.5 m.

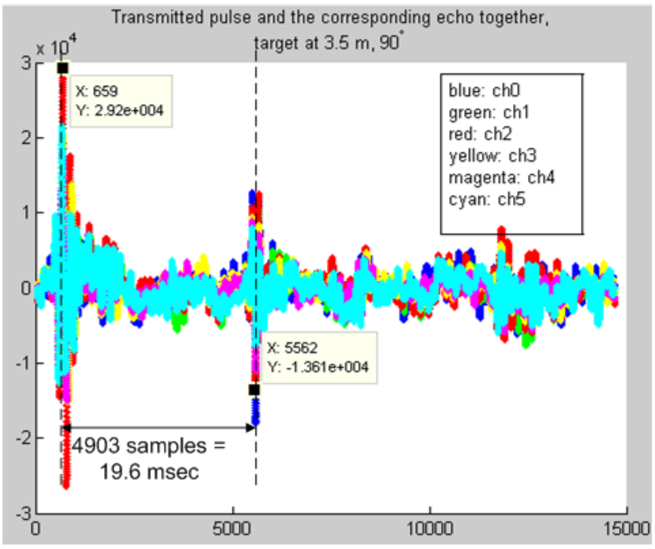


Figure 5.17: Entire transmit-receive cycle captured, target position: 3.5 m, 90°

Beamformer Output Observation: The next action that the *Acoustic Radar System Controller* takes after capturing the channel waveforms is to apply delay-and-sum beamforming for the current steer angle by running the *Receive Delay-and-Sum Beamformer* module (see Section 4.2.6). The beamformer output, which is nothing but the receive BRAM content after beamforming is done is given below in Figure 5.18. There exist 2 echo pulses in the beamformer output, one is generated by our target which is placed in a distance of 3.5 m and a direction of 90°, and the other is due to the reflection from the windows at the very end of the room in which the acoustic radar tests are conducted. The first 499 samples of the capture given in Figure 5.18 are all zeros. Note that 499 is not an arbitrary number, it is 1 sample less than the length of the acoustic radar pulse, which is 2 msec or in number of samples, (2 msec)/(4 uses per sample) = 500 samples. These zeros are inserted intentionally for the operation of the following module, which is *Correlator*. The peak of the beamformer output is at the 5140th index of the Receive BRAM. The effective index of the peak is actually 250 sample (it is half the length of the acoustical radar pulse) less than this value. So, the effective peak is reached after first 5140-250 = 4890 samples, or after (4890 samples) $\times$ (4 usec/sample) = 19.6 msec, which is exactly consistent with the result calculated above.

Note that acoustic radar system has a parameter called *number of captures per angle* (refer the explanation of *number_of_captures_per_angle_reg* in Section 4.2.10.1). From Figure 5.19,

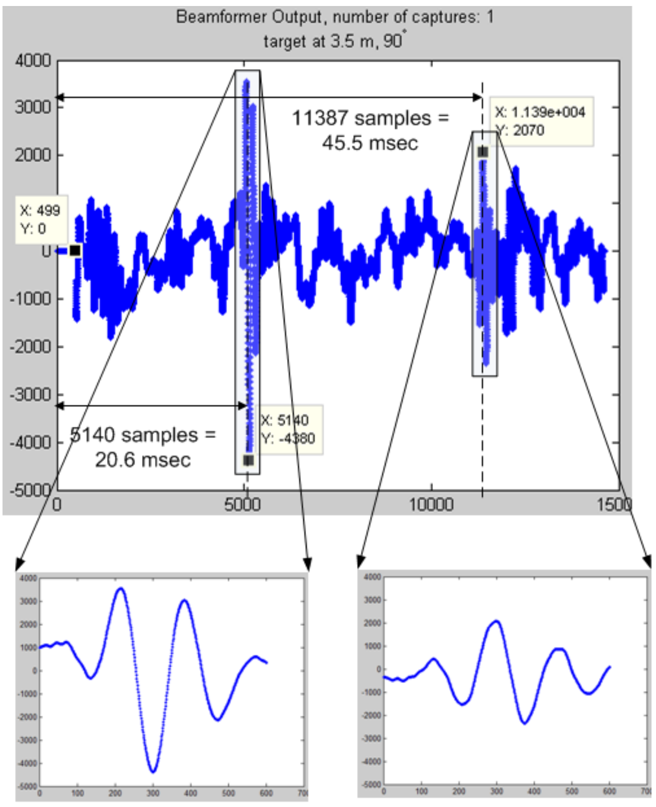


Figure 5.18: Beamformer Output, number of captures: 1, target position: 3.5 m, 90°

it is obvious that the magnitude of the beamformer output is directly proportional (actually the magnitude depends linearly on the number of captures parameter) to the *number of captures per angle* parameter. One more point to emphasize about this parameter is that it affects the noise level (and obviously SNR) of the beamformer output. When the *number of captures* is increased, the resulting beamformer output which is calculated over multiple captures becomes smoother, the noise components are suppressed more (this is called **Noise Suppression**). This effect can be observed from Figure 5.20 (this figure is obtained via zooming into an arbitrary portion -where acoustic radar pulse is not present- of the beamformer output).

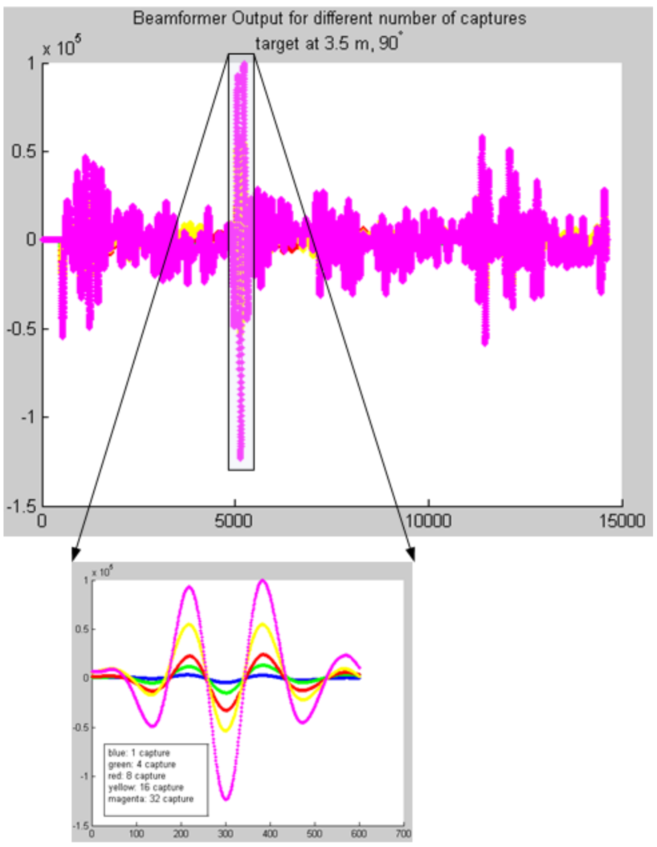


Figure 5.19: Beamformer Output for different values of *number of captures per angle* parameter, target position: 3.5 m, 90°

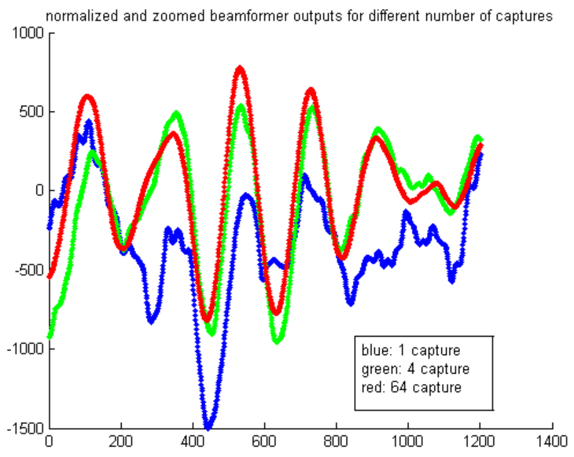


Figure 5.20: Normalized beamformer outputs zoomed, *number of captures* parameter varied, target position: 3.5 m, 90°

Correlator Output Observation: The next action after delay-and-sum beamforming is to apply cross-correlation to the beamformed signal. It is achieved by *Correlator* module (see Section 4.2.7). The index of the maximum value of the correlator output is the enough information to locate the target in the acoustic radar system.

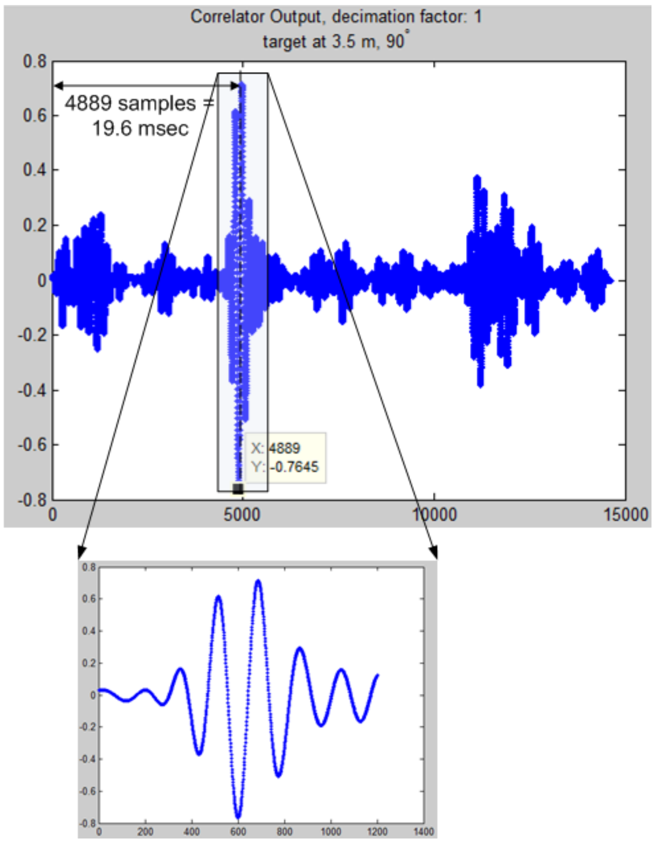


Figure 5.21: Correlator Output, decimation factor: 1, target position: 3.5 m, 90°

In the acoustical radar system, cross correlation operation has an important parameter called *correlation decimation factor*, which is explained in detail in Section 4.2.7. *Correlator decimation factor* is an important parameter for speed. The correlation output with the decimation factor chosen as 2 is given in Figure 5.22. Note from the figure that there are zeros between the correlation samples. It is because that this capture is the entire receive BRAM content after cross-correlation is applied to the signal. *Correlator* module writes the correlation result to the BRAM by skipping a deterministic number of entries. Number of entries skipped is 1 less the *correlator decimation factor* parameter (which is 1 for the case in Figure 5.22).

The cross-correlation result is reliable for a relatively high number of *decimation factor*s. For

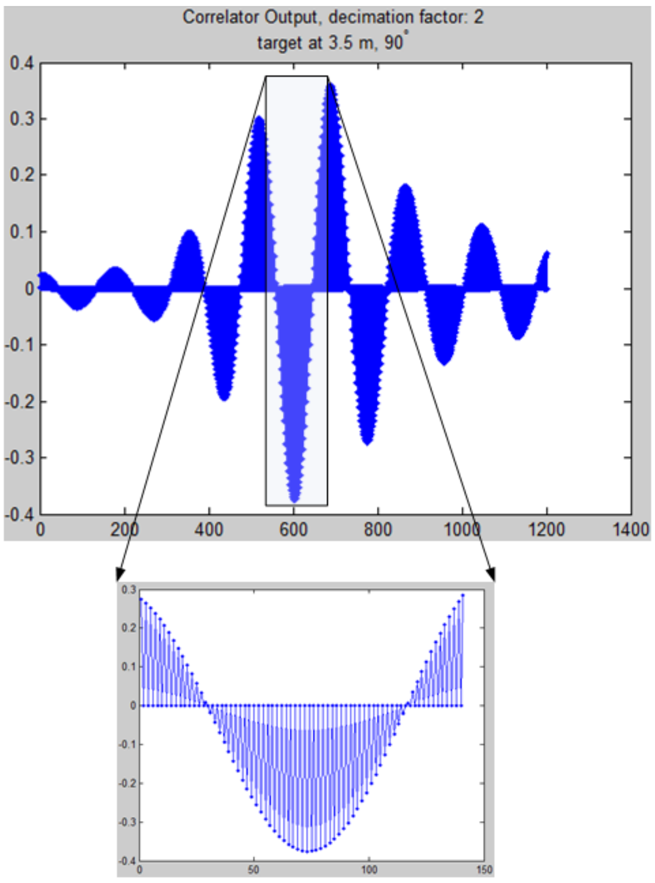


Figure 5.22: Correlator Output, decimation factor: 2, target position: 3.5 m, 90°

example, even for a decimation of 32, for which the correlation result is given in Figure 5.23, the system works well. This observation is obviously related to the length of the acoustic radar pulse, which is 500 samples. So, for the *decimation factor* of 32, picking 500/32 ≅ 15 samples from the acoustical radar pulse to be used in cross-correlation calculation is still enough to get a satisfying result. The magnitude of the cross-correlation depends on the *decimation factor* parameter, it is obviously inversely proportional to the *decimation factor*, as seen from Figure 5.24.

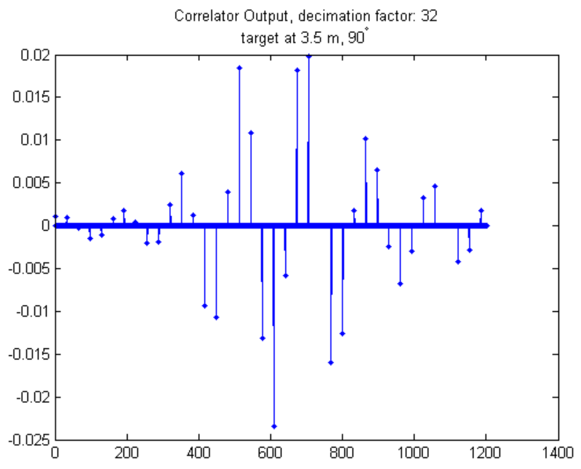


Figure 5.23: Correlator Output, decimation factor: 32, target position: 3.5 m, 90°

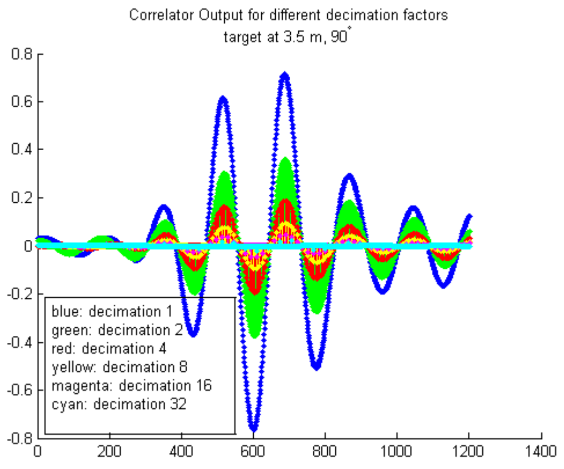
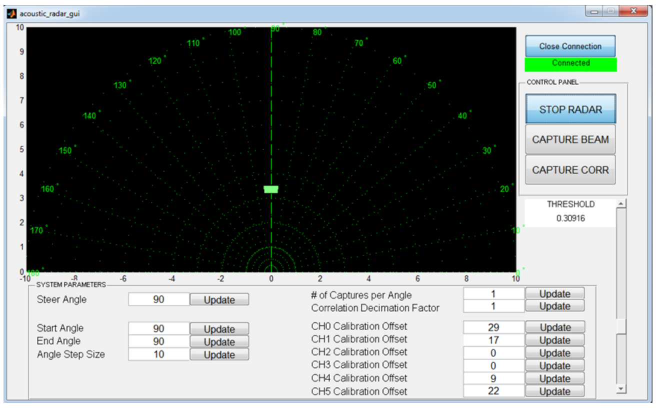


Figure 5.24: Correlator Output for different values of *correlator decimation factor* parameter, target position: 3.5 m, 90°

Radar Display Observation: For the current scenario, a screenshot taken from the radar display is given below in Figure 5.25. The position of the displayed target is as expected, at 3.5 m, 90°. Note that, this graphical user interface, which is called *System Configuration and Display* unit, is not only used to display the detected targets. As seen from Figure 5.25, it provides user the ability to modify many system parameters like *start angle*, *end angle*, *correlation decimation factor*, etc. The threshold value which is used in target detection decision is set from this GUI as well.



THRESHOLD 0.30916

SYSTEM PARAMETERS

Steer Angle	90
Start Angle	90
End Angle	90
Angle Step Size	10
# of Captures per Angle	1
Correlation Decimation Factor	1
CH0 Calibration Offset	29
CH1 Calibration Offset	17
CH2 Calibration Offset	0
CH3 Calibration Offset	0
CH4 Calibration Offset	9
CH5 Calibration Offset	22

Figure 5.25: Radar screenshot, target position: 3.5 m, 90°

Scenario 2

In this part, the very same steps will be followed with Scenario 1 (which are *transmitted pulse observation*, *echo pulse observation*, *beamformer output observation*, *correlator output observation* and *radar display observation*). So, the details explained above *will not be repeated* also in here.

Transmitted Pulse Observation: Unlike the previous test in which the target is placed to the direction of 90°, there is phase difference between channel waveforms both for transmission and reception.

Note that for 60° steer angle, the delay between successive channels is calculated as 173.35 μsec. In Figure 5.27, the phase diffence is noted as 228 usec. It is not close to the theoretical result. But note that, as mentioned above, the transmitted pulse observation is not that perfect because of the near field effect.

Note also that the leading channel among the transmit array channels is channel 5 (Figure 5.27), which is an obvious requirement to steer the sound to the direction of 60°.

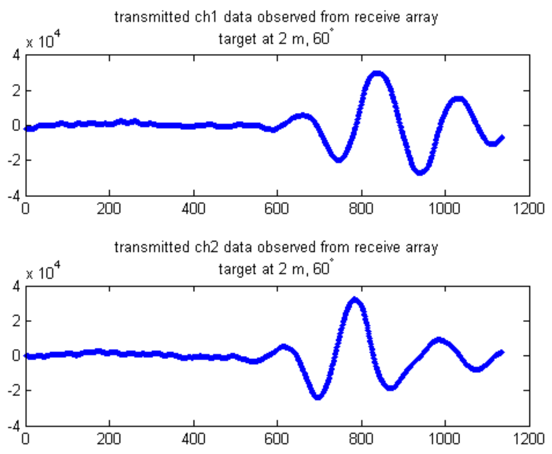


Figure 5.26: Transmitted CH1 and CH2 waveforms from the receive array's point of view, note the phase difference between channels, target position: 2 m, 60°

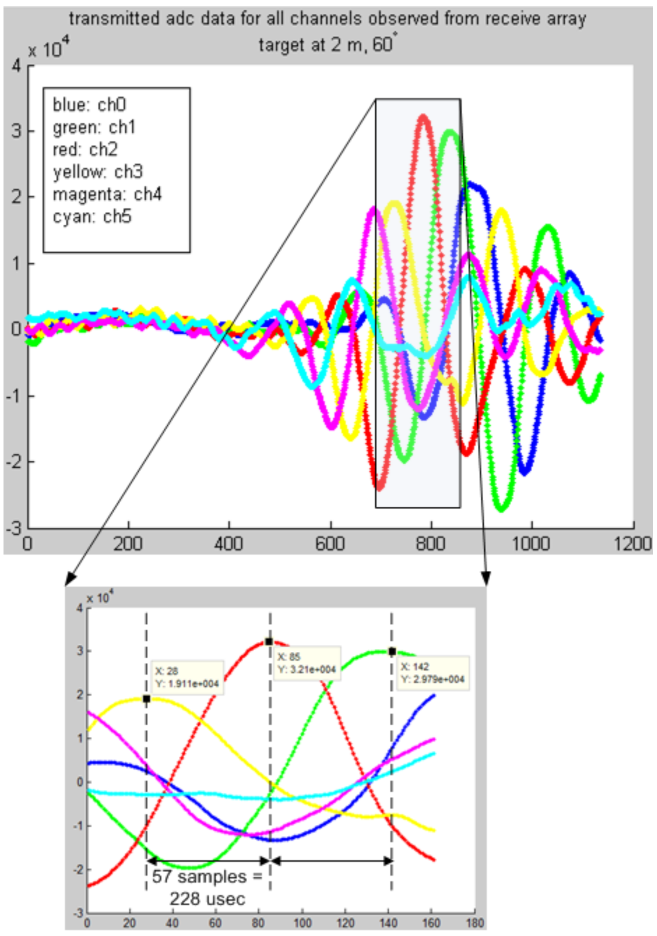


Figure 5.27: Transmitted waveforms for all channels from the receive array's point of view, note the phase difference between channels, target position: 3.5 m, 90°

Echo Pulse Observation: The received echo observed in two successive channels and in all channels are given below in figures 5.28 and 5.29, respectively. Note the phase difference between successive channels. It is 200 usec, close to the theoretical expectation, which is calculated as 173.35 usec. Also note that the leading channel in the receive array is the channel 0 (it was channel 5 for the transmit array) as expected.

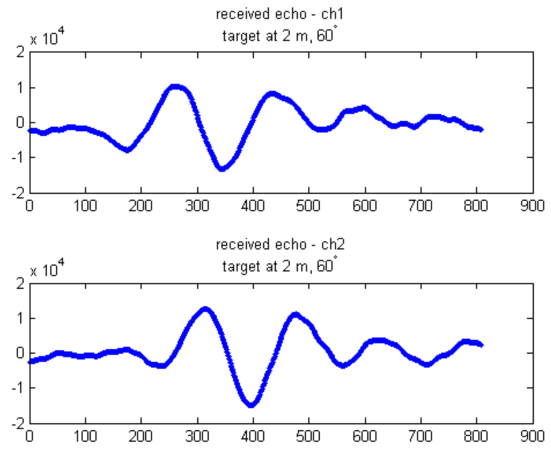


Figure 5.28: Received echo from CH1 and CH2, note the phase difference, target position: 2 m, 60°

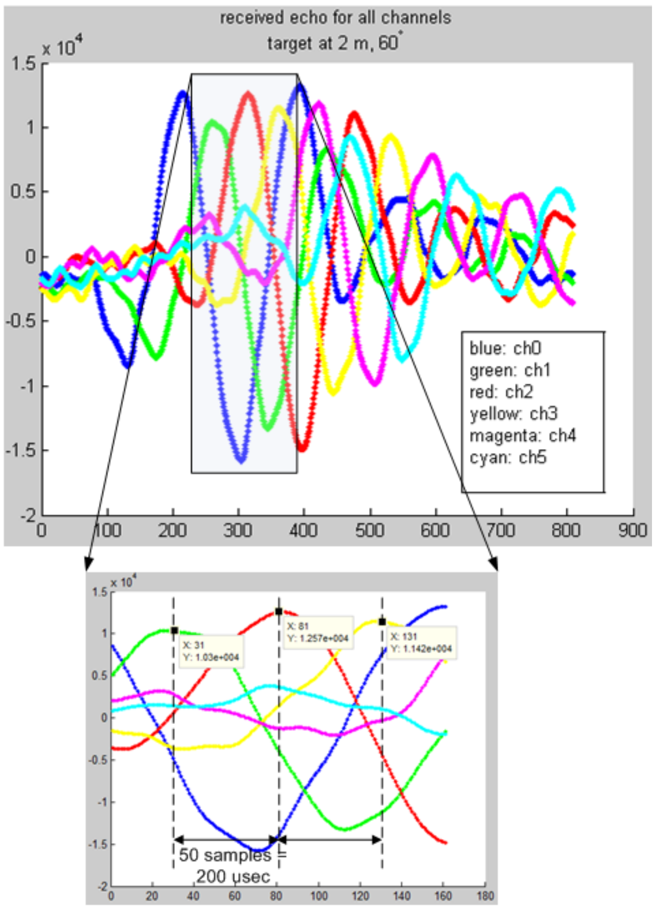


Figure 5.29: Received echo from all channels, no phase difference observed, target position: 2 m, 60°

The transmitted and received pulses are given above in separate plots (in figures 5.27 and 5.29) from the receive array's point of view. The entire transmit-receive cycle is given below in Figure 5.30. Note that the time of flight of the transmitted radar pulse in air is 11.2 milliseconds as given in the figure. It corresponds to a distance of 11.2 msec×346.13 m/sec = 3.8753 m (speed of sound at 25° temperature = 346.13 m/sec). So, it means that the target is at 3.8753/2 = 1.9377 ~ 2 m.

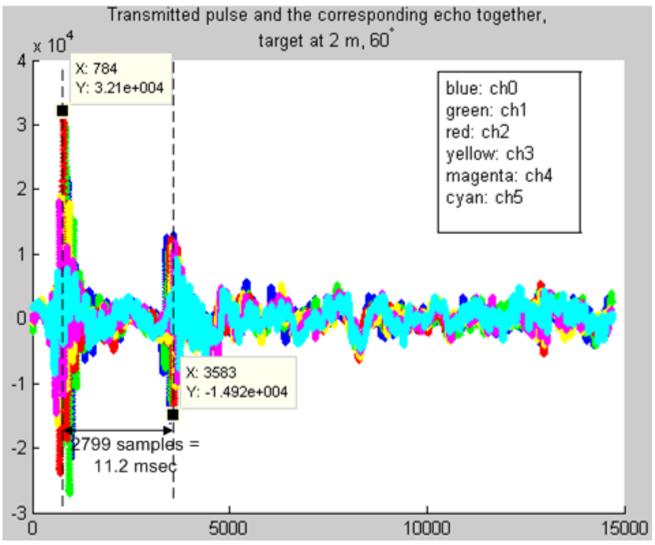


Figure 5.30: Entire transmit-receive cycle captured, target position: 2 m, 60°

Beamformer Output Observation: Note from Figure 5.31 that the peak of the beamformer output is at the 3073th index of the Receive BRAM. The effective index of the peak is actually 250 sample (it is half the length of the acoustical radar pulse) less than this value. So, the effective peak is reached after first 3073-250 = 2823 samples, or after (2823 sample) × (4 usec/sample) = 11.3 msec, which is very close to the result calculated above.

The beamformer output for different *number of captures* parameter and the observation of this parameter to the SNR of the beamformed signal is given in figures 5.32 and 5.33, respectively. Refer the explanations given in scenario 1.

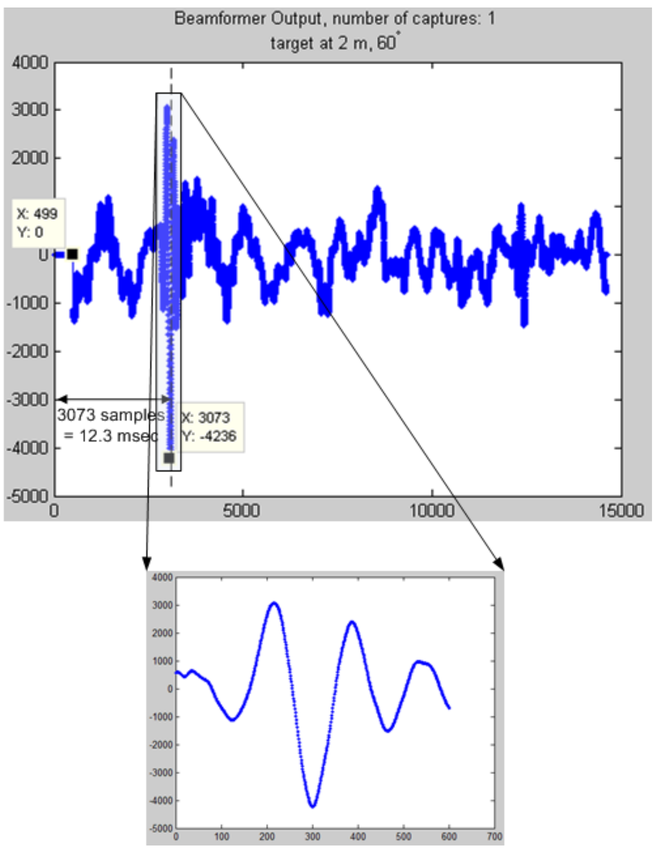


Figure 5.31: Beamformer Output, number of captures: 1, target position: 2 m, 60°

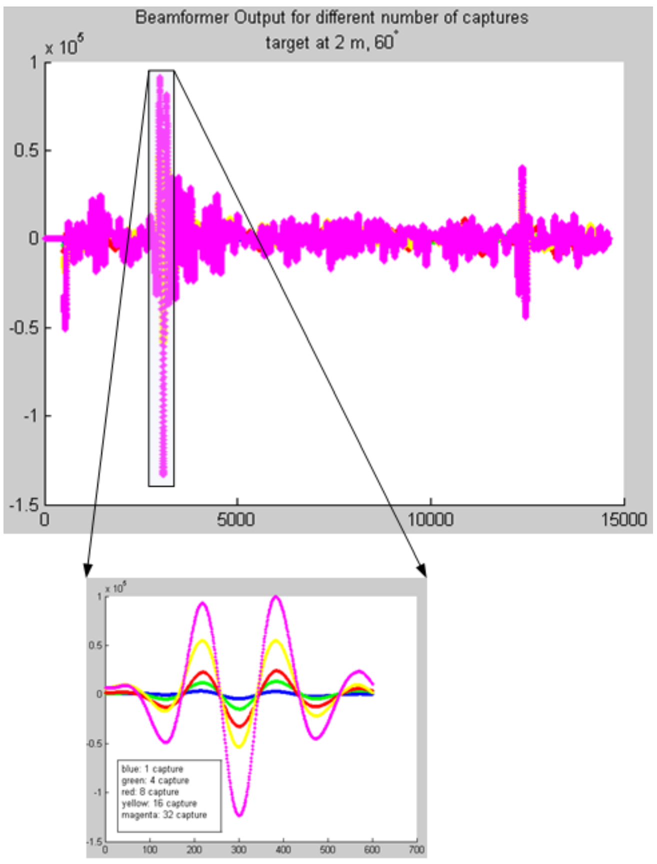


Figure 5.32: Beamformer Output for different values of *number of captures per angle* parameter, target position: 2 m, 60°

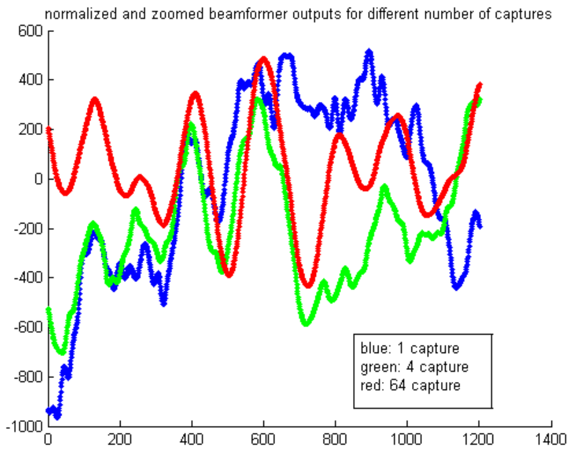


Figure 5.33: Normalized beamformer outputs zoomed, *number of captures* parameter varied, target position: 2 m, 60°

Correlator Output Observation: The results for the current scenario in which the target is placed at a distance of 2 m and the direction of 60° is given. Refer the explanations given in scenario 1.

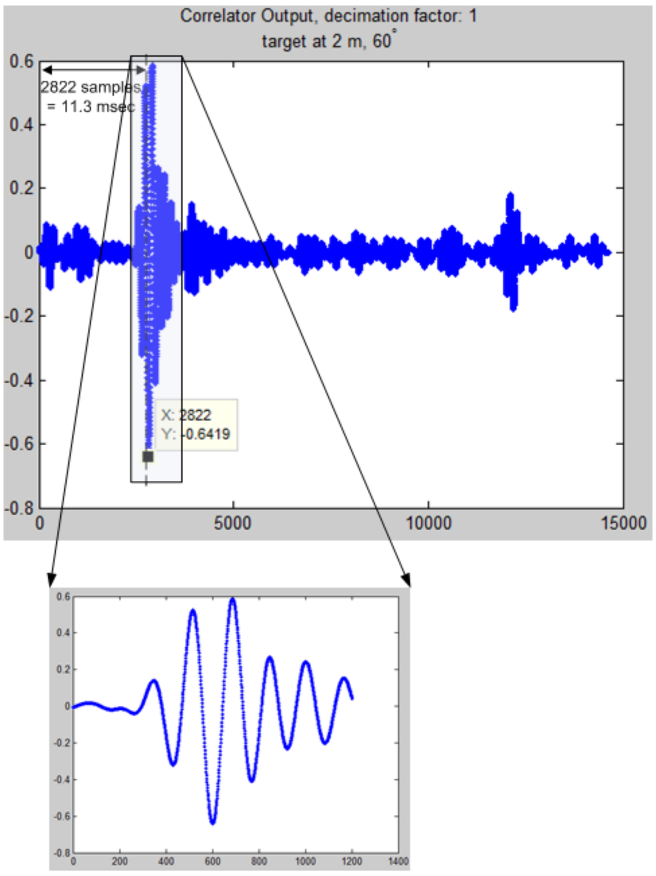


Figure 5.34: Correlator Output, decimation factor: 1, target position: 2 m, 60°

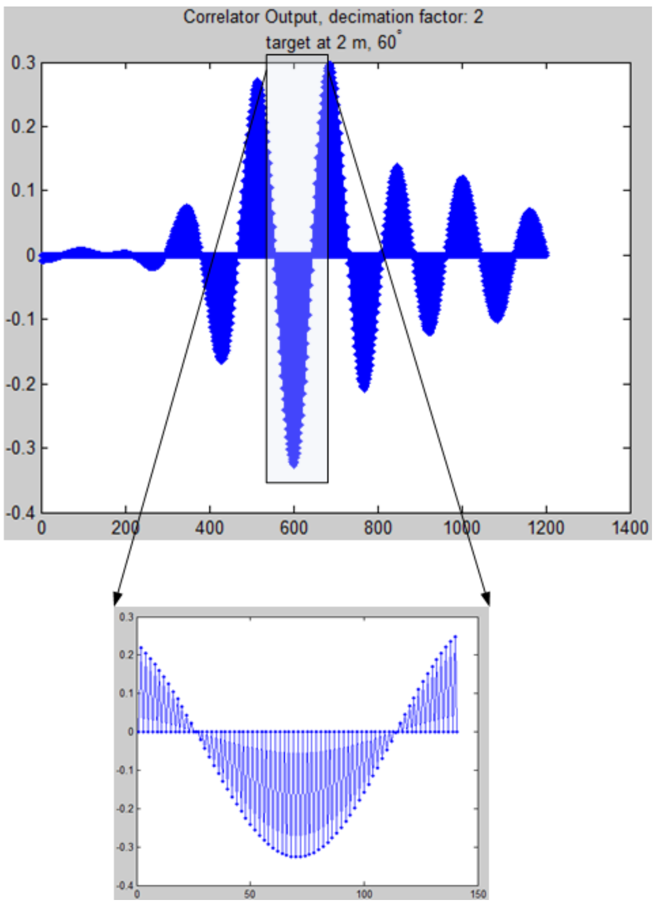


Figure 5.35: Correlator Output, decimation factor: 2, target position: 2 m, 60°

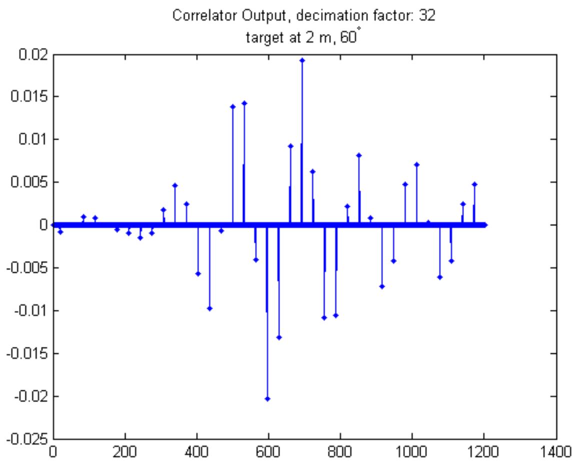


Figure 5.36: Correlator Output, decimation factor: 32, target position: 2 m, 60°

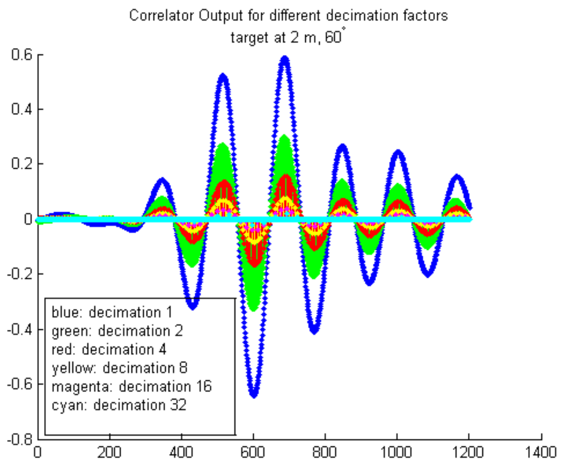
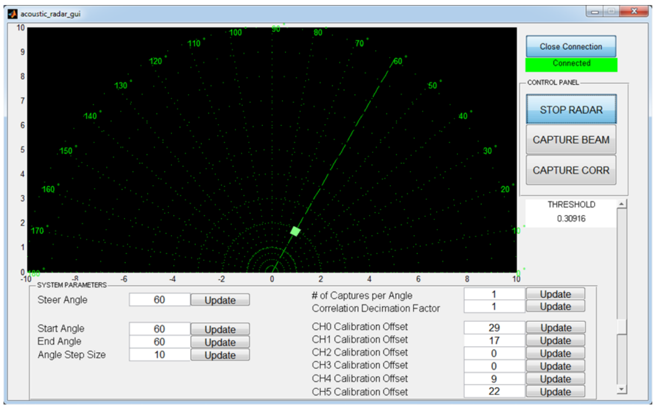


Figure 5.37: Correlator Output for different values of *correlator decimation factor* parameter, target position: 2 m, 60°

Radar Display Observation: For the current scenario, a screenshot taken from the radar display is given below in Figure 5.38. The position of the displayed target is as expected, at 2 m, 60°. Refer the explanations given in scenario 1.



THRESHOLD
0.30916

Steer Angle	60
Start Angle	60
End Angle	60
Angle Step Size	10
# of Captures per Angle	1
Correlation Decimation Factor	1
CH0 Calibration Offset	29
CH1 Calibration Offset	17
CH2 Calibration Offset	0
CH3 Calibration Offset	0
CH4 Calibration Offset	9
CH5 Calibration Offset	22

Figure 5.38: Radar screenshot, target position: 2 m, 60°

In addition to the data captures obtained by the hardware experiments explained above, a demo video is recorded. In this demo video, firstly, the acoustic radar system is described. Then, the system is recorded on video during radar operation, in which the target is moved towards/from the tx/rx array in different directions.

CHAPTER 6

CONCLUSIONS

In this thesis work, an acoustic radar system is implemented real-time on Field Programmable Gate Arrays. The implemented acoustic radar system employs 6-element, *linear* transmit (loudspeakers) and receive (microphones) arrays. The operation of the system mainly relies on *delay-and-sum beamforming*. This algorithm is used both for steering the radar pulse to a certain direction, which is *transmit beamforming*, and listening to that particular direction in order to capture the corresponding echoes, which is *receive beamforming*.

The operation in an acoustical environment has some advantages and disadvantages over that of electromagnetic waves. When compared with electromagnetic waves, the attenuation on acoustic waves is higher which limits the maximum operating range of the system considerably. In addition, electrical to acoustic energy conversion is inefficient. The efficiency is usually around 2.5% for common loudspeakers. Also multipath effect is more apparent in an acoustical system due to the nature of the sound waves. However, robustness to multipath signals can be noted as the implemented acoustic radar system uses beam steering. Another disadvantage of the implemented system may be noted as the limitation of the performance by physical structure of the arrays. In order to have a narrower bandwidth and also to avoid spatial aliasing at the same time, the inter-element distance of the transmit/receive arrays must be chosen as $\lambda/2$. The minimum inter-element distance achieved in the system -due to the dimensions of the loudspeakers in the transmit array- is 0.12 cm. One of the advantages of our acoustic radar system is related with the transmit and receive arrays. In an electromagnetic system, there is mutual coupling between antennas in an array. This degrades the performance and periodic calibration is required for proper operation. In an acoustical system, there is no interaction between array elements. Implementation of an acoustical system is easier and cheaper from that of an electromagnetic one which requires the use of expensive and carefully tuned devices.

The implemented system is not only a radar device which just displays the detected targets on a screen, but also a real-time testbed to observe the performance of utilized signal processing algorithms on a real acoustical environment. Starting from the front-end of the system (raw data captured from analog-to-digital converters), it is possible to observe all steps of signal processing throughout the system. Also, a number of parameters (*number of captures* and *cross-correlation decimation factor* parameters are the most important ones) are defined which provides the user the ability to observe the effect of different system configurations on overall performance. In an implementational point of view, the system has another feature such that it utilizes very special and optimized elements of the field programmable gate arrays. Filter taps are implemented using specialized high-speed DSP primitives. Also, in low-pass filtering of 6-channels and calculating the cross-correlation between the received waveform and the radar pulse, some innovative approaches are developed. Namely, only one filter structure which works in a multiplexed manner is used to filter 6 channels. This results in a great saving of FPGA resources. In cross-correlation calculation, instead of employing a filter with high number of taps, a method which uses only one filter tap -a multiply-accumulate structure- is developed. In terms of resource utilization, this method is a good choice. However, processing time that this method requires is considerably long. The shared-memory utilization of the system must also be pointed out which results in great savings of FPGA memory resources.

The implemented acoustic radar system has a highly modular and systematical structure. The operation of the system is shared between a number of specialized units under the management of a high level controller module. This structure makes the system very flexible. It may be developed further easily based on the current infrastructure. In this work, a very basic beamformer, which is the delay-and-sum beamformer is implemented. As a future work, more sophisticated algorithms (like *adaptive* beamforming techniques) may be implemented. Another future approach may be using the FPGA design only for interfacing the computer with the transmit and receive arrays. So very complex algorithms may be implemented and evaluated on computer using real acoustical environment data, which is prescious. As a result, acoustic radar system provides a very flexible and reliable infrastructure for future studies.

REFERENCES

[1] Fourikis, Nicholas. Advanced Array Systems, Applications and RF Technologies. Academic Press, 2000

[2] How Far Off Is That German Gun ? How 63 German guns were located by sound waves alone in single day, Popular Science monthly, December 1918, page 39

[3] Antoniou Ioannis, Jørgensen, Hans E., Ormel, Frank, Bradley, Stuart, von Hünerbein, Sabine, Emeis, Stefan, Warmbier, Günter, On the Theory of SODAR Measurement Techniques, Final Reporting on WP1, EU WISE Project NNE5-2001-297, Risø National Laboratory, Roskilde, Denmark, April 2003

[4] Antoniou Ioannis, Jørgensen, Hans E., Ormel, Frank, Bradley, Stuart, von Hünerbein, Sabine, Emeis, Stefan, Warmbier, Günter, On the Theory of SODAR Measurement Techniques, Final Reporting on WP1, EU WISE Project NNE5-2001-297, Risø National Laboratory, Roskilde, Denmark, April 2003

[5] Gao, Robert X., A Dynamic Ultrasonic Ranging System As a Mobility Aid for the Blind, Engineering in Medicine and Biology Society, 1995, IEEE 17th Annual Conference, Vol 2, pp. 1631-1632, 1995

[6] Shoval, S., Borenstein, J., Koren, Y., Navbelt and the guide-cane (obstacle-avoidance systems for the blind and visually impaired), IEEE Robot. Autom. Mag., Vol. 10, No 1., pp. 9-20, 2003

[7] Strakowski, M., Kosmowski, B., Kowalik, R., and Wierzba P., An ultrasonic obstacle detector based on phase beamforming principles, IEEE Sensors J., Vol 6, No 1., pp 179-186, 2006

[8] Harput, Sevan, Bozkurt, Ayhan, Ultrasonic Phased Array Device for Acoustic Imaging in Air, IEEE Sensors J. Vol 8., No 11, pp. 1755-1762, 2008.

[9] C.A. Balanis. Antenna Theory Analysis and Design. John Wiley and Sons Inc., 1997

[10] Hamish Meikle, Modern Radar Systems, Second Edition, Artech House Inc., 2008

[11] Prabhakar S. Naidu, Sensor Array Signal Processing, Second Edition, CRC Press Taylor & Francis Group LLC, 2009

[12] Stoica, Petre; Moses, Randolph L., Spectral Analysis of Signals, Prentice Hall Inc., 2005

[13] Van Veen, Barry D., Buckley, Kevin M, Beamforming, A Versatile Approach to Spatial Filtering, IEEE ASSP Magazine, 4-24, April 1988.

[14] Von Ramm, Olaf T., Smith, Stephen W, Beam Steering with Linear Arrays, IEEE Transactions on Biomedical Engineering, Vol. BME-30, No. 8, August 1983

[15] Benesty, Jacob; Chen, Jingdong; Huang, Yiteng; Microphone Array Signal Processing, Springer 2008

[16] Ridenour, Louis N., Clarke, Avis M., Radar System Engineering, McGraw Hill Book Company, Inc. 1947.

[17] Levanon, Nadav, Radar Principles, John Wiley and Sons, 1988

[18] Levanon, Nadav, Radar Signals, John Wiley and Sons, 2004

[19] Spartan-6 Family Overview, XILINX Inc.

[20] ATLYS Board Reference Manual, DIGILENT Inc.

[21] DEM-DAI1602 Evaluation Fixture User Guide, Texas Instruments Inc.

[22] ADS8364 Evaluation Module User Guide, Texas Instruments Inc.

[23] WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores, Revision B.3, OpenCores Organization, www.opencores.org

[24] Distributed Memory Generator v6.2 , XILINX Inc.

[25] Spartan-6 FPGA Block RAM Resources User Guide, XILINX Inc.

[26] LogiCORE IP Block Memory Generator v6.1, XILINX Inc.

[27] PCM1602, 24 Bit, 192 kHz Sampling, 6 Channel, Enhanced Mutilevel, Delta-Sigma Digital-to-Analog Converter, Texas Instruments Inc.

[28] LogiCORE IP FIFO Generator v8.2, XILINX Inc.

[29] Chapman, Ken, Digitally Removing a DC Offset: DSP Without Mathematics, White Paper: Xilinx FPGAs, WP279(v1.0), July 18, 2008,

[30] Function: *fdesign*; Signal Processing Toolbox, Analog and Digital Filters, Digital Filter Design, *www.mathworks.com/help/signal/ref/fdesign.html#brygv68-376*

[31] Spartan-6 FPGA DSP48A1 Slice User Guide, UG389 (v1.1), August 13, 2009, XILINX Inc.

[32] 250 kBPS, 16-Bit, 6-Channel, Simultaneous Sampling ADC Datasheet, Texas Instruments Inc.

[33] Spartan-6 FPGA Clocking Sources User Guide, XILINX Inc.

[34] LogiCORE IP Tri-Mode Ethernet MAC v4.5 User Guide, XILINX Inc.

[35] LogiCORE IP Tri-Mode Ethernet MAC v4.5 Product Specification, XILINX Inc.

[36] 88E1111 Integrated 10/100/1000 Ultra Gigabit Ethernet Transceiver Datasheet, Marvell Inc.

[37] ChipScope Pro Software and Cores User Guide,UG029, July 6, 2011, XILINX Inc.

[38] ModelSim-PE Student Edition 2007-2012 Mentor Graphics Corporation.

APPENDIX A

MODULE PORT DESCRIPTIONS

A.1 TRANSMIT PULSE GENERATOR PORT DESCRIPTIONS

Name	Direction	Description
WishBone Master Interface		
CLK_I	IN	Clock: All the internal operation is synchronous to this signal
RST_I	IN	Reset: this is an active high synchronous reset input, the finite state machine and all internal registers are initialized when this signal is asserted.
WBS_ADR_I [1:0]	IN	WishBone Slave Address [23] For now, there is only one register inside the module, which is Steer Angle Register. When master intends to write to this register, it must set the value of this port to “00“. “00“ → Steer Angle Register address
WBS_DAT_I [7:0]	IN	Wishbone Slave Input Data [23] Steer Angle information is supplied to the module via this port. It may take values [0, 180].
WBS_WE_I	IN	WishBone Slave Write Enable [23] ‘1‘ → WishBone Write Access ‘0‘ → WishBone Read Access

WBS_STB_I	IN	WishBone Slave Strobe [23] Master initializes a write/read access via asserting this signal, all other WishBone signals is meaningful in the presence of this signal.
WBS_DAT_O [7:0]	OUT	WishBone Slave Output Data [23] When WBM (WishBone Master) performs a read access to a WBS register, data is supplied to it via this port.
WBS_ACK_O	OUT	WishBone Slave Acknowledge [23] Implies that the write/read access initialized by WBM is accepted by WBS.
Data Output Interface		
DATA_VALID	OUT	Only when this signal is high, the output signal CHX_DATA is meaningful, otherwise, their values should be considered as “don‘t care“.
CHX_DATA [23:0]	OUT	Generated data for channel X, (X may take values 0, 1, 2, 3, 4, 5, there are six channels)

Table A.1: Transmit Pulse Generator Module Port Description

A.2 PCM1602 DAC CONTROLLER PORT DESCRIPTIONS

Name	**Direction**	**Description**
Input Data Interface		
CLK	IN	Internal FIFO Write Domain Clock. This clock is directly connected to Write Clock of the channel FIFOs. It has nothing to do with the internal operation of the DAC Controller module, DAC Controller operation is synchronous to the FIFO read clock, which is BCLK.
RST	IN	(Active High) Synchronous reset input, channel FIFOs and all internal registers are initialized if this signal is asserted.

BCLK	IN	Module clock input. This is the main clock input of the DAC Controller module; all processes are synchronous to this clock. Obviously, this is directly connected to the Read Clock of internal channel FIFOs. DAC audio serial interface is driven with this clock.
FULLx	OUT	(Active High) Directly connected to FULL port of the internal channel x FIFO (x ∈ [0, 5]). If this is high, it implies that channel x FIFO is full and write accesses will be unsuccessful.
WR_ENx	IN	(Active High) Directly connected to WR_EN port of channel x FIFO. A write access to FIFO is performed by asserting this signal.
DINx	IN	Directly connected to DIN port of channel x FIFO. This is nothing but the data that is intended to be written to the channel x FIFO.
	PCM1602 DAC Audio Serial Interface [27]	
LRCK	OUT	Serial audio left/right clock. When this is high, left channel data (channels 0, 2 and 4) is present in DATA0, DATA1 and DATA2 ports. When this is low, right channel data (channels 1, 3 and 5) is present in DATA0, DATA1 and DATA2 ports. This clock is equal to the sampling rate, $f_s = 52083$.
BCK	OUT	Shift clock input for serial audio data. It is 48×fs in the default 24-Bit Right Justified format. This is generated via dividing the 5 MHz BCLK input of the module by 2.
DATA0	OUT	Serial audio data input for channel 0 and 1 (refer LRCK port explanation)
DATA1	OUT	Serial audio data input for channel 2 and 3 (refer LRCK port explanation)

DATA2	OUT	Serial audio data input for channel 4 and 5 (refer LRCK port explanation)
SCLK	OUT	System Clock Input of the PCM1602, input frequency may be 128f$_s$, 192f$_s$, 256f$_s$, 384f$_s$, 512f$_s$, or 768f$_s$. In the particular implementation it is chosen as 384f$_s$, which is 20 MHz.

Table A.2: PCM1602 DAC Controller Port Description

A.3 ADS8364 ADC CONTROLLER PORT DESCRIPTIONS

Name	**Direction**	**Description**
Clock and Reset Interface		
CLK_I	IN	CLK input (5 MHz in the particular implementation, 250 kSPS sampling is achieved by this clock), directly connected to CLK input of ADS8364 ADC chip. All internal signals are synchronous to this clock.
RST_I	IN	(Active High) It is used as the reset for internal logic and its inverse is directly connected to RESET# or ADS8364 chip.
ADS8364 ADC Interface [32]		
CLK	OUT	Clock signal for ADS8364. Maximum clock frequency is 5 MHz. (In the particular implementation, it is chosen such to achieve 250 kSPS sampling rate.)
RESET_N	OUT	Active Low reset signal for ADS8364
HOLD_A_N HOLD_B_N HOLD_C_N	OUT	HOLD signals for each channel pair. A conversion is initiated on the ADS8364 by bringing the HOLD_X_N pin LOW for a minimum of 20 ns. In ADS8364_ADC_CTRL, these 3 signals tied together. (because ADC Controller module operates in CYCLE MODE [32])

ADDRESS[2:0]	OUT	Channel Address. 000 → CHA0 is to be read 001 → CHA1 is to be read 010 → CHB0 is to be read 011 → CHB1 is to be read 100 → CHC0 is to be read 101 → CHC1 is to be read 110 → CYCLE_MODE reads registers CHA0 through CHC1 on successive transitions of the read line. to be read 111 → FIFO Mode The particular impementation uses CYCLE MODE
ADD	OUT	Active High. In cycle and FIFO modes, if this signal is asserted, address information is captured from the chip with 16 bit output data. Tied to ground, no address information is requested from the ADS8364 Chip
BYTE	OUT	Active High. If there is only an 8 bit bus available on the board, MS bits (15:8) and LS bits (7:0) are read in successive RD_N assertions if BYTE is high. Tied to ground. ADC Controller module uses 16-bit data bus, not 8-bit.
RD_N	OUT	Active Low. Read enable signal. To get data from the data bus of the chip, this signal must be asserted.
WR_N	OUT	Active Low. Write enable signal. It is used if ADS8364 is software controlled. In ADS8364, it is tied high (deasserted), chip is hardware controlled.
CS_N	OUT	Active Low. Chip Select signal. RD_N is valid only when this signal is low.
FDATA	IN	Active High. First Data Indicator. Not used in ADS8364_ADC_CTRL

EOC_N	IN	Active Low. End of Conversion signal. Asserted when new data of the internal ADC is latched into the output registers, usually happens 16.5 clock cycles after HOLD_X_N is asserted.
DATA[15:0]	IN	ADS8364 Data Bus. When RD_N and CS_N is low simultaneously, the data corresponding to the channel for which a conversion is initiated before appears on this bus.
Channel FIFO Interface		
WE_X	OUT	Active Write Enable signal for the output fifo of channel X
DATA_X [15:0]	OUT	Data to be written to the output fifo of channel X.

Table A.3: ADS8364 ADC Controller Port Description

A.4 DC OFFSET REMOVER PORT DESCRIPTIONS

Name	**Direction**	**Description**
CLK	IN	Clock: All the internal operation is synchronous to this signal. The period of this signal is nothing but the ΔT of the digital RC Circuit
RST	IN	Reset: All internal registers are initialized when this signal is asserted
VIN [15:0]	IN	Input data whose DC offset is intended to be removed
VIN_VALID	IN	This signal qualifies the input data VIN. The input value is ignored when VIN_VALID is low
VOUT [15:0]	OUT	Output waveform without DC offset
DC_OFFSET [15:0]	OUT	Calculated DC offset

VOUT_VALID	OUT	This signal qualifies the output data. VOUT and DC_OFFSET values are valid only when this signal is high

Table A.4: DC Offset Remover Port Description

A.5 6-CHANNEL FIR LOW-PASS FILTER PORT DESCRIPTIONS

Name	**Direction**	**Description**
CLK	IN	Clock: all the internal operation is synchronous to this signal.
RST	IN	Reset: Active high synchronous reset input, all internal registers are initialized when this signal is asserted
Data Interface		
DIN_CHX [15:0]	IN	Channel X input data, $X \in \{0, 1, 2, 3, 4, 5\}$, there are 6 input channels
DIN_CHX_VALID	IN	DIN_CHX valid signal. Data present in port DIN_CHX is only taken into account when this signal is high; otherwise, it is ignored.
DOUT_CHX [15:0]	OUT	Low-pass filtered Channel X output data, $X \in \{0, 1, 2, 3, 4, 5\}$, there are 6 input channels
DOUT_CHX_VALID	OUT	DOUT_CHX valid signal. Data present in port DOUT_CHX must only be taken into account when this signal is high; otherwise, it should be ignored.

Table A.5: 6-Channel FIR Low-Pass Filter Module Port Description

A.6 RECEIVE DELAY AND SUM BEAMFORMER PORT DESCRIPTIONS

Name	**Direction**	**Description**

WBS Interface		
CLK_I	IN	Clock: all the internal operation is synchronous to this signal.
RST_I	IN	Reset: Active high synchronous reset input, all internal registers and finite state machine are initialized when this signal is asserted
WBS_ADR_I [7:0]	IN	WishBone Slave Register Address [23] x"00" → *rxbf_ch0_calibration_offset_reg* register address x"01" → *rxbf_ch1_calibration_offset_reg* register address x"02" → *rxbf_ch2_calibration_offset_reg* register address x"03" → *rxbf_ch3_calibration_offset_reg* register address x"04" → *rxbf_ch4_calibration_offset_reg* register address x"05" → *rxbf_ch5_calibration_offset_reg* register address x"06" → *rxbf_steer_angle_reg* register address x"07" → *rxbf_status_reg* register address
WBS_DAT_I [7:0]	IN	WishBone Slave Input Data [23] Calibration offset registers may take any values in range [0, 255]. *rxbf_steer_angle_reg* may take values in range [0, 180] with a resolution of 1 degrees.
WBS_WE_I	IN	WishBone Slave Write Enable [23] '1' → WishBone Write Access '0' → WishBone Read Access
WBS_STB_I	IN	WishBone Slave Strobe [23] Master initializes a write/read access via asserting this signal, all other WishBone signals are meaningful in the presence of this signal.

WBS_DAT_O [7:0]	OUT	WishBone Slave Output Data [23] When WBM (WishBone Master) performs a read access to a WBS register, data is supplied to it via this port. In this module there is just one read accessible register, which is *rxbf_status_reg*. The possible values *rxbf_status_reg* are as follows: x“00“ → beamformer idle x“01“ → beamformer done
WBS_ACK_O	OUT	WishBone Slave Acknowledge [23] Implies that the write/read access initialized by WBM is accepted by WBS.
	Channel FIFOs Read Interface	
EMPTY_FROM_ CHANNELx_FIFO	IN	Empty signal for the channel x ($x \in [0,5]$) ‘1‘ → FIFO is empty ‘0‘ → FIFO is *not* empty
RD_EN_TO _CHx_FIFO	OUT	Active high read enable signal for channel x ($x \in [0,5]$) FIFO.
DOUT_FROM _CHx_FIFO [15:0]	OUT	Data read from channel x ($x \in [0,5]$) FIFO
	Receive BRAM Interface	
EN_TO_BRAM	OUT	BRAM Active High Enable Write/Read accesses to BRAM is only valid when this enable signal is high, otherwise, the accesses are ignored by BRAM
WE_TO_BRAM	OUT	BRAM Write Enable ‘1‘ → Write Access to BRAM ‘0‘ → Read Access to BRAM
ADDR_TO_BRAM [13:0]	OUT	BRAM Address
DIN_TO_BRAM [35:0]	OUT	BRAM Data Input

DOUT_FROM_BRAMIN [35:0]		BRAM Data Output

Table A.6: Receive Delay and Sum Beamformer Port Description

A.7 CORRELATOR PORT DESCRIPTIONS

Name	Direction	Description
WBS Interface		
CLK_I	IN	Clock: all the internal operation is synchronous to this signal.
RST_I	IN	Reset: Active high synchronous reset input, all internal registers and finite state machine are initialized when this signal is asserted
WBS_ADR_I [7:0]	IN	WishBone Slave Register Address [23] x“00“ $\rightarrow$ *corr_dec_factor_reg* register address x“01“ $\rightarrow$ *corr_status_reg* register address x“02“ $\rightarrow$ *corr_max_index1_reg* register address x“03“ $\rightarrow$ *corr_max_index0_reg* register address x“04“ $\rightarrow$ *corr_max_value3_reg* register address x“05“ $\rightarrow$ *corr_max_value2_reg* register address x“06“ $\rightarrow$ *corr_max_value1_reg* register address x“07“ $\rightarrow$ *corr_max_value0_reg* register address
WBS_DAT_I [7:0]	IN	WishBone Slave Input Data [23] *corr_dec_factor_reg* register may take any values in range [1, 255]
WBS_WE_I	IN	WishBone Slave Write Enable [23] ‘1‘ $\rightarrow$ WishBone Write Access ‘0‘ $\rightarrow$ WishBone Read Access

WBS_STB_I	IN	WishBone Slave Strobe [23] Master initializes a write/read access via asserting this signal, all other WishBone signals are meaningful in the presence of this signal.
WBS_DAT_O [7:0]	OUT	WishBone Slave Output Data [23] When WBM (WishBone Master) performs a read access to a WBS register, data is supplied to it via this port. In this module there is just one read accessible register, which is *rxbf_status_reg*. The possible values *rxbf_status_reg* are as follows: x“00“ → correlator idle x“01“ → correlator done Data read from other read accessible register may be explained as follows: *corr_max_index1_reg* → bits [15:8] of correlation maximum value index *corr_max_index0_reg* → bits [7:0] of correlation maximum value index *corr_max_value3_reg* → bits [31:24] of correlation maximum value *corr_max_value2_reg* → bits [23:16] of correlation maximum value *corr_max_value1_reg* → bits [15:8] of correlation maximum value *corr_max_value0_reg* → bits [7:0] of correlation maximum value
WBS_ACK_O	OUT	WishBone Slave Acknowledge [23] Implies that the write/read access initialized by WBM is accepted by WBS.
Receive BRAM Interface		

EN_TO_BRAM	OUT	BRAM Active High Enable Write/Read accesses to BRAM is only valid when this enable signal is high, otherwise, the accesses are ignored by BRAM
WE_TO_BRAM	OUT	BRAM Write Enable ‘1‘ → Write Access to BRAM ‘0‘ → Read Access to BRAM
ADDR_TO_BRAM [13:0]	OUT	BRAM Address
DIN_TO_BRAM [35:0]	OUT	BRAM Data Input
DOUT_FROM_BRAM [35:0]	IN	BRAM Data Output

Table A.7: Correlator Port Description

A.8 CLOCK GENERATOR PORT DESCRIPTIONS

Name	Direction	Description
Input Clock and Reset Interface		
CLK_100MHZ _FROM_BOARD	IN	Input clock which is sourced from the 100 MHz CMOS oscillator mounted on ATLYS FPGA Board. This is connected to PLL input clock port and all clocks are generated from this.
RST	IN	(Active High) Asynchronous reset to PLL. In the particular implementation, this is Hard Reset, namely, it is directly connected to the switch which is present on the ATLYS FPGA Board.
Output Clock and Reset Interface		
CLK_100MHZ	OUT	100 MHz clock output.
CLK_20MHZ	OUT	20 MHz clock output.
CLK_5MHZ	OUT	5 MHz clock output.

PLL_LOCKED	OUT	(Active High). Asynchronous output from the PLL that provides the user with an indication that PLL has achieved phase alignment and is ready for operation. In the particular implementation, this signal is used as global reset for the Acoustic Radar System.

Table A.8: Clock Generator Port Description

A.9 ETHERNET CONTROLLER PORT DESCRIPTIONS

Name	**Direction**	**Description**
RX WishBone Slave PortRX WishBone Slave Port		
Receive WishBone accessible registers are the following: *rx_eth_header_reg*: this register holds header fields of the expected Ethernet frame. Receive FSM filters the received Ethernet packets with the information recorded in this register. *rx_ip_header_reg*: this register holds header fields of the expected IP packet Receive FSM filters the received IP packets with the information recorded in this register. *rx_udp_header_reg*: this register holds header fields of the expected UDP packet. Receive FSM filters the received UDP packets with the information recorded in this register. *rx_data_register*: the payload of the received UDP packet is recorded in register. WishBone Master gets the packet via reading this register		
Transmit WishBone Slave Interface		
tx_eth_header_reg: this register holds header fields of the generated Ethernet frames. Transmit FSM forms the Ethernet Header field of the packet with the information recorded in this register *tx_ip_header_reg*: this register holds header fields of the generated IP packets. Transmit FSM forms the IP Header field of the packet with the information recorded in this register		

tx_udp_header_reg: this register holds header fields of the generated UDP packets. Transmit FSM forms the UDP Header field of the packet with the information recorded in this register *tx_instruction_register*: WishBone Master initiates a transmit process via writing this register. Writing this register causes Transmit FSM to be triggered to form a UDP packet. *tx_data_register*: WishBone Master writes the data to this register to be transmitted. Transmit FSM puts the content of this register to the data field of the transmitted UDP packet.		
MII Interface [36]		
COL	IN	MII Collision Signal. In full duplex mode, it is always low and not used by the Ethernet Controller Module
CRS	IN	MII Carrier Sense. It is asserted by the PHY Transceiver when the receive medium is non-idle.
TXD[3:0]	OUT	MII Transmit Data. It is the data to be transmitted onto the cable.
TX_EN	OUT	MII Transmit Enable. When asserted, data on TXD[3:0] along with TX_ER is encoded and transmitted onto the cable
TX_ER	OUT	MII Transmit Error. When TX_ER and TX_EN are both asserted, the transmit error symbol is transmitted onto the cable. When TX_ER is asserted with TX_EN deasserted, carrier extension symbol is transmitted onto the cable.
TX_CLK	IN	MII Transmit Clock. It provides a 25 MHz clock reference for TX_EN, TX_ER and TXD[3:0].
RXD[3:0]	IN	MII Receive Data. Symbols received on the cable are decoded and presented on RXD[3:0].
RX_DV	IN	MII Receive Data Valid. When asserted, data received on the cable is decoded and presented on RXD[3:0] and RX_ER.

RX_ER	IN	MII Receive Error. When RX_ER and RX_DV are both asserted, the signals indicate an error symbol is detected on the cable. When RX_ERR is asserted with RX_DV deasserted, a false carrier or carrier extension symbol is detected on the cable.
RX_CLK	IN	MII Receive Clock. It provides a 25 MHz clock reference for RX_DV, RX_ER and RXD[3:0]

Table A.9: Ethernet Controller Port Description