IMPLEMENTATION OF A DIRECTION FINDING ALGORITHM ON AN FPGA PLATFORM

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

BY

ABDULLAH VOLKAN İPEK

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

OCTOBER 2006

Approval of the Graduate School of (Name of the Graduate School)

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. İsmet ERKMEN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Mete SEVERCAN
Supervisor

Examining Committee Members (first name belongs to the chairperson of the jury and the second name belongs to supervisor)

Prof. Dr. Yalçın TANIK	(METU, EE)	_____
Prof. Dr. Mete SEVERCAN	(METU, EE)	_____
Assoc. Prof. Dr. Sencer KOÇ	(METU, EE)	_____
Assist. Prof. Dr. Çağatay CANDAN	(METU, EE)	_____
Ülkü DOYURAN	(ASELSAN)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Abdullah Volkan İPEK

Signature :

ABSTRACT

IMPLEMENTATION OF A DIRECTION FINDING ALGORITHM
ON AN FPGA PLATFORM

İPEK, Abdullah Volkan

M.S., Department of Electrical and Electronics Engineering

Supervisor : Prof. Dr. Mete SEVERCAN

October 2006, 83 pages

In this thesis work, the implementations of the monopulse amplitude comparison and phase comparison DF algorithms are performed on an FPGA platform. After the mathematical formulation of the algorithms using maximum-likelihood approach is done, software simulations are carried out to validate and find the DF accuracies of the algorithms under various conditions. Then the algorithms are implemented on an FPGA platform by utilizing platform specific software tools. Block diagrams of the hardware implementations are given and explained in detail. Then simulations of hardware implementation of both algorithms are performed. Using the results of the simulations, DF accuracies under certain conditions are evaluated and compared to software simulations results.

Keywords: Direction Finding, Amplitude Comparison, Phase Comparison, Interferometer, FPGA

ÖZ

YÖN BULMA ALGORİTMASININ FPGA PLATFORMUNDA UYGULANMASI

İPEK, Abdullah Volkan
Yüksek Lisans Tezi, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi : Prof. Dr. Mete SEVERCAN

Ekim 2006, 83 sayfa

Bu tez çalışmasında, genlik karşılaştırmalı ve faz karşılaştırmalı yön bulma algoritmaları FPGA platformunda gerçeklenmiştir. Maksimum olabilirlik yaklaşımı kullanılarak algoritma matematiksel olarak formüle edildikten sonra, yazılım benzetimleri yapılarak algoritma geçerli kılınmış, farklı durumlarda algoritmaların yön bulma doğrulukları bulunmuştur. Daha sonra platforma özel yazılım araçları kullanılarak algoritmalar FPGA platformunda uygulanmıştır. Donanım uygulamalarının blok şemaları verilerek detaylı bir şekilde anlatılmıştır. Her iki algoritmanın donanım uygulamasının benzetimi yapılmıştır. Benzetim çalışmalarının sonuçları kullanılarak belirli koşullar altındaki yön bulma doğruluğu değerlendirilmiş, yazılım benzetim sonuçları ile karşılaştırılmıştır.

Anahtar Kelimeler: Yön Bulma, Genlik Karşılaştırma, Faz Karşılaştırma, FPGA

To My Parents and To My Lovely Sister

ACKNOWLEDGMENTS

I would like to thank Prof. Dr. Mete Severcan for his valuable supervision, support and tolerance throughout the development and improvement of this thesis.

I am grateful to Turgut Çelikadam, Metin Şengül and Serkan Sevim for their support throughout the development and the improvement of this thesis. I am also grateful to Aselsan Electronics Industries Inc. for the resources and facilities that I use throughout thesis.

Thanks a lot to all my friends for their great encouragement and their valuable help to accomplish this work.

Lastly, I would like to thank my parents for bringing up and trusting in me, and Demet Salman, for giving me the strength and courage to finish this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AC : Amplitude Comparison

ADC : Analog to Digital Converter

AOA : Angle of Arrival

CW : Continuous Wave

DF : Direction Finding

DOA : Direction of Arrival

DSP : Digital Signal Processing

ESM : Electronic Support Measures

FIR : Finite Impulse Response

FPGA : Field Programmable Gate Arrays

GSPS : Giga Samples per Second

LPF : Low Pass Filter

MPPS : Mega Pulse per Second

MSPS : Mega Samples per Second

PC : Phase Comparison

PDW : Pulse Descriptor Word

PRI : Pulse Repetition Interval

RF : Radio Frequency

RMS : Root Mean Square

RWR : Radar Warning Receivers

SNR : Signal to Noise Ratio

TOA : Time of Arrival

UCA : Uniform Circular Array

ULA : Uniform Line Array

CHAPTER I

INTRODUCTION

1.1 Direction Finding

The ultimate aim for the Electronic Support Measurement (ESM) system is to identify the RF guided weapon. The general procedure of the identification against RF guided weapons can be summarized as follows. First the ESM system intercepts the radiations associated with that weapon, and then separates these signals from other intercepted signals. Then the parameters of the selected signals are measured. ESM system compares the parameters against a stored set of parameters to identify the type of the sensor. Finally the weapon is identified by previously identified definitions of the weapons. In the process of identifying the RF guided weapon, the ESM system provides information on the angular direction of the intercepted signal [1]. A direction finder is a passive device that determines the direction/angle of arrival (DOA/AOA) of radio-frequency energy [2]. Determining the DOA or AOA of signal is fundamental to ESM systems, since emitters cannot change on pulse to pulse basis.

Purpose of direction finding operations is to detect the position of the emitter, which can be calculated from the bearings of several direction finders. There are many factors that should be considered in direction finding problem. Choice of the array geometry and antennas, the receiver structure and the optimum algorithm that is employed can be considered as main factors. Obviously the interactions between these parts must not be ignored.

The antenna pattern and polarization are important parameters that affect the DF accuracy of the system. In DF systems, multiple antennas are arranged in various geometrical configurations. Array geometry affects various aspects of the DF system such as resolution, sensitivity to system errors and the type of processing used to

produce DOA estimates. In most arrays, the elements of an array are identical, which is often more convenient, simpler, and more practical. The most commonly considered structures are the linear and circular arrays. Antenna spacing over the specified geometry can be selected to be uniform or non-uniform. Uniform Line Array (ULA) is implemented with inter-element spacing less than or equal to $\lambda/2$; where λ is the wavelength at the center frequency. The importance of non-uniform line array structures is that they are considerably more robust as compared to ULA's with respect to their radiation characteristics. A linear array does not provide uniform resolution capability over the entire horizon. One of the main advantages of a circular array over linear arrays is its 360º azimuthal coverage which is very important especially in ESM applications. The structure which the elements are arranged uniformly around 360º in azimuth is called as a uniform circular array (UCA). UCA geometry is one of the most commonly used array geometry in DF systems [3].

There are various approaches to estimate the angular location of the emitter. The most commonly used methods are monopulse and high resolution methods. High resolution methods are more complex compared to monopulse. They are based on correlation matrix, eigenvalue and eigenspace computations. Computational costs of these methods are higher, which is an important drawback for many practical systems. On the other side, monopulse systems should operate in real time, provide fast responses, and be able to process as large as pulses per second to find the AOA.

1.2 The meaning of Monopulse

It is stated in [4] that the word *monopulse* was first introduced by H.T. Budenbom of the Bell Telephone Laboratories in 1946. It is a hybrid word, because mono meaning one comes from Greek, while pulse comes from Latin. It becomes very common because it clearly expressed the ability to collect, from each pulse, the information needed for a pair of two coordinate angle estimates, whereas the older angle-sensing techniques of conical scan and lobe switching required several (at least four) pulses to form a pair of angle estimates.

The term *simultaneous lobing*, synonymous with monopulse, is a more accurate description of the technique, although it is less commonly used. It is the simultaneity rather than "one pulse" that is essential. As a matter of fact, most monopulse radars do not extract angle estimates from each pulse. The usual practice is to smooth or integrate the raw single-pulse signals over some interval before forming the angle estimates. Furthermore, monopulse is not confined to pulsed radars. It can be used just as well in Continuous Wave (CW) or modulated CW radars. It can be used passive mode to track a source of signals or noise such as transmitting antenna, a jammer or the sun [4].

1.3 Monopulse DF Systems

The two primary techniques used for monopulse direction finding are the amplitude-comparison method and the interferometer or phase comparison method. The phase-comparison method generally has the advantage of greater accuracy, but the amplitude-comparison method is used extensively due to its lower complexity and cost. In the following chapters, information about amplitude and phase comparison systems will be given.

1.3.1 Amplitude Comparison DF Systems

Virtually all currently deployed radar warning receiving (RWR) systems use amplitude-comparison direction finding. A basic amplitude-comparison receiver derives a ratio, and ultimately angle-of-arrival or bearing, from a pair of independent receiving channels, which utilize squinted antenna elements that are usually equidistantly spaced to provide an instantaneous 360º coverage. Typically, four or six antenna elements and receiver channels are used in such systems, and wideband logarithmic video detectors provide the signals for bearing-angle determination. The monopulse ratio is obtained by subtraction of the detected logarithmic signals, and the bearing is computed from the value of the ratio [1].

In amplitude comparison, typically broadband spiral antenna elements whose patterns can be approximated by Gaussian-shaped beams are used. Gaussian-shaped

beams have the property that the logarithmic output ratio slope in dB is linear as a function of angle of arrival. Thus, a digital look-up table can be used to determine the angle directly. However, both the antenna beamwidth and squint angle vary with frequency over the multi-octave bands used in RWRs. Pattern shape variations cause a larger pattern crossover loss for high frequencies and reduced slope sensitivity at low frequencies. Partial compensation of these effects, including antenna squint, can be implemented using a look-up table if frequency information is available in the RWR. Otherwise, gross compensation can be made, depending upon the RF octave band utilized [1].

It is stated in [5] that the typical accuracies can be expected to range from 3 to 10 degrees RMS for multi-octave frequency band amplitude-comparison systems which cover 360 degrees with four to six antennas. The four-quadrant amplitude-comparison DF systems employed in RWRs have the advantage of simplicity, reliability, and low cost. Usually, only one antenna per quadrant is employed which covers the 2 to 18 GHz band. The disadvantages are poor accuracy and sensitivity, which result from the broad-beam antennas employed. Both accuracy and sensitivity can be improved by expanding the number of antennas employed. For example, expanding to eight antennas would double the accuracy and provide 3 dB more gain. As the number of antennas increases, it becomes appropriate to consider multiple-beam-forming antennas rather than just increasing the number of individual antennas.

1.3.2 Interferometer DF Techniques

Interferometer can be considered as specific cases of antenna arrays. All antenna elements center may lie in a straight line at equal spacing (ULA) or at different spacing to obtain time of arrival (TOA) difference of the incoming waves. The difference of TOA will lead to a measurable phase difference, which is used for determination of the angle of arrival (AOA) information. The concept of using phase difference of the received signals can be used with circular array structures to cover 360 degrees azimuth from antennas mounted at a point.

Interferometer system antennas typically use broad antenna beams with beamwidths of the order 90 degrees. It is stated in [1] that the lack of directivity produces several deleterious effects. First, it limits the system sensitivity due to reduced antenna gain. Second, it open the system to interference signals often include multipath from strong signals which can limit the accuracy of the interferometer.

An interferometer system consisting of two antennas in space causes an ambiguity in determination of AOA. The system has coverage of 180 degrees in azimuth; however it is not clear from which half of the hemisphere the signal originated. Practical interferometer systems solve this problem by using another system such as an amplitude monopulse DF system to select proper estimate of DOA or use quadrant arrays of antennas shielded from each other, or a circular array of monopulses to cover a 360 degree field of view [6]. Another method to resolve ambiguities is to use a multiple antenna elements, called multiple baseline interferometers. In a typical design of multiple baseline interferometers, there exists a reference antenna and a series of companion antennas, spaced in line and located at different distances from the reference antenna in order to operate together.

In multiple baseline interferometer systems, there are two types of choice of antenna element location. A harmonic binary related interferometer system divides each aperture baseline by a factor of 2 is given in Figure 1.1.

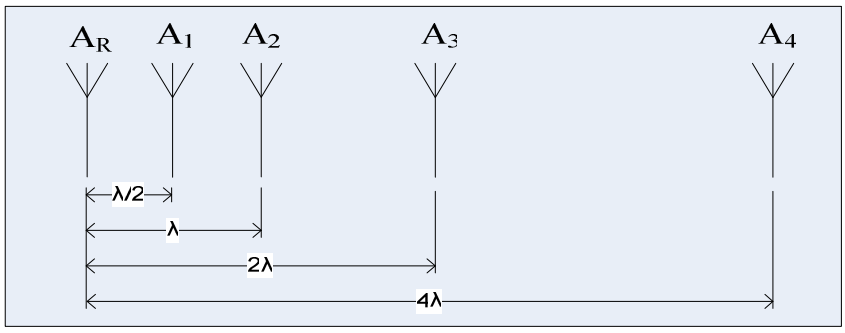

Figure 1.1 Harmonic Binary Related Interferometer

The number of ambiguities is increased as the spacing to the reference antenna is increased. In the selected quadrant, first antenna pair, forming a $\lambda/2$ baseline, solves the ambiguity problem and determines the AOA, and then the next pair improves the resolution and achieves a better determination. The process continues up to longest baseline.

In non-harmonic related spaced antennas, the antenna element spacing, which is not necessarily $\lambda/2$, is determined up to frequency to solve the ambiguity problem. In this configuration, ambiguities will be presented at many frequencies. Nevertheless, knowledge of spacing or spacing ratios and the frequency of the incoming signal and utilization of look up tables and some logic algorithms, the angle of arrival can be determined.

The non-harmonic related spacing is more preferred to harmonic-binary related spacing antennas, since it has a wide RF bandwidth of coverage. And the binary system is limited at the high-end frequency range to half-wavelength baseline spacing. Also it is stated in [3] that the performance of uniform and different non-uniform LA structures are compared, and in the event of single target in additive white Gaussian noise, the non uniform arrays found to provide significant improvement over arrays of the same number of elements which shows the importance of array geometry.

When high DF accuracy is needed in an interferometer system, alternatively antenna baseline spacing can be increased. The increased baseline spacing results in multi-lobe structures along the coverage range, which requires less signal to noise ratio (SNR) to achieve same the DF accuracy. For a two element interferometer with 16λ spacing would end up 33 lobes through ±90 degree azimuth coverage. Within the 33 lobes, it would only require a SNR of 20dB to achieve 0.1 degree accuracy [1]. Although there are 33 ambiguous regions on the coverage range, ambiguity will be resolved by employing a third antenna element with $\lambda/2$ baseline spacing. At the defined SNR value, it will provide a DF accuracy of 3 degrees, which is sufficient to solve the ambiguity [1].

1.4 Description of the Thesis

In this work, two algorithms based on monopulse DF techniques, namely amplitude comparison and phase comparison, to solve the problem of estimating direction of arrival (DOA) of plane waves impinging on the antennas from unknown direction are employed. The sources and the antennas are assumed to be coplanar, and therefore, the problem is constrained to estimate the DOA's in azimuth only. Amplitude comparison and phase interferometer techniques are studied in this work by employing maximum likelihood estimation approach. The algorithms are implemented on an FPGA platform and simulation results are obtained by varying parameters such as SNR, baseline spacing etc. The implementation details of the design, discussions and comparisons of the simulation results of the algorithms are explained in detail.

1.5 Outline of the Thesis

This thesis is organized as follows: In Chapter II, the problem is stated and formulated; the amplitude comparison and phase interferometer DF algorithms based on ML estimation are developed and described in detail.

In Chapter III, in order to investigate the performance of the proposed algorithms, the root-mean-square (RMS) errors at different angles and various conditions are calculated. Typical results of the software simulations are presented, along with the discussions related to the effects of different parameters. Some practical limitations, assumptions, modifications and practical advantages related to the implementation of this algorithm on an FPGA platform are also discussed. The results of the implementation are compared with the expected values.

Finally, both monopulse DF algorithms are summarized. Moreover some concluding remarks of these algorithms and hardware implementations are discussed in Chapter IV.

CHAPTER II

DF ALGORITHMS

2.1 Amplitude Comparison Algorithm

The DF system considered in this part consists of a circular array of N equally spaced antennas. It is assumed that there is a single transmitter and the signal arrives at the antenna at angle of θ. The geometry of the antennas (for N=6) is given in Figure 2.1.

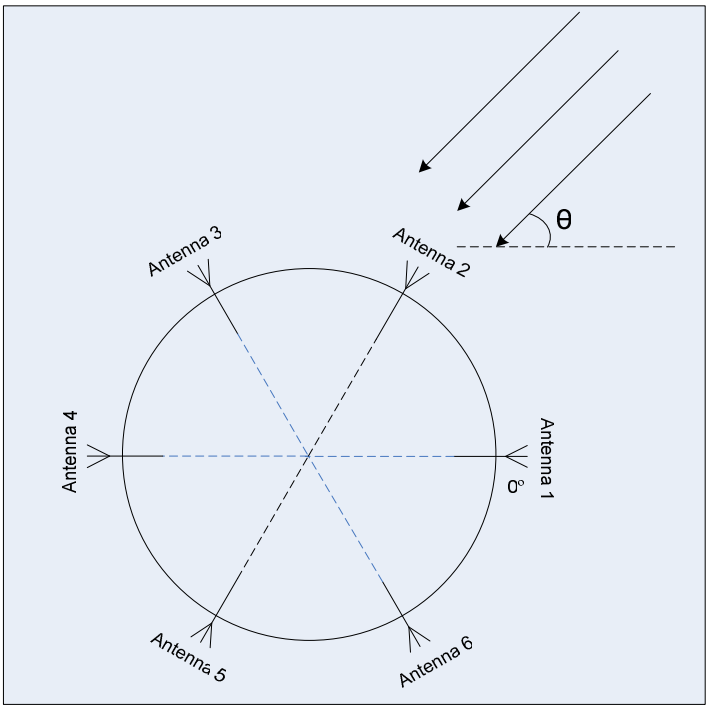

Figure 2.1 Circular Array Geometry (N=6)

The received signal is defined by

$$s_i = AR_i(\theta - \theta_i) + n_i \quad \textbf{(2-1)}$$

where θ and A represents AOA and amplitude of the incoming signal respectively. $R_i(\ )$ denotes the power pattern of the antenna, θ_i is the angular position of the i^{th} (i=1,2…N) antenna, which is given by $(2\pi / N)(i-1)$ and n_i is the receiver noise.

In this case, the problem is to estimate θ by using observations of the received signals s_i's. At that point, two assumptions are made. First assumption is that the additive noise is assumed to be white Gaussian noise of variance σ^2. Second one is that the additive noise of each receiver channel is independent of each other. The Gaussian distributed random variable having zero mean and a variance of σ^2 has the density function of the form

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \quad \textbf{(2-2)}$$

Since random variables are independent from each other, their joint density function is the product of the densities for each Gaussian variable. The probability density function (pdf) can be expressed as

$$f(A,\theta) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(s_i - AR_i(\theta-\theta_i))^2}{2\sigma^2}} \quad \textbf{(2-3)}$$

The maximum likelihood estimation for parameter θ is the estimate, which maximizes the joint density function.

$$\theta = \arg\max_{\theta}(f(A,\theta)) \quad \textbf{(2-4)}$$

Maximizing the maximum likelihood function is equivalent to minimizing the negative of logarithm of it, since the maximum likelihood function is a monotonic function. By eliminating constant terms, the equation can be rewritten as

$$J(A,\theta) = \sum_{i=0}^{N-1} (s_i - AR_i(\theta - \theta_i))^2 \quad \textbf{(2-5)}$$

Therefore the estimate θ can be obtained by minimizing J as

$$\theta = \arg\min_{\theta}\left(J(A,\theta)\right) \tag{2-6}$$

The maximum likelihood estimation is function of A and θ. In order to find the minimum point of the function, a derivative with respect to A is taken and set to zero. Taking derivative of Eq.(2-5) with respect to A yields to

$$\frac{\partial J(A,\theta)}{\partial A} = \sum_{i=0}^{N-1}\left(-2s_i R_i(\theta-\theta_i) + 2AR_i^2(\theta-\theta_i)\right) = 0 \tag{2-7}$$

$$\hat{A} = \frac{\sum_{i=0}^{N-1}\left(s_i R_i(\theta-\theta_i)\right)}{\sum_{i=0}^{N-1} R_i^2(\theta-\theta_i)} \tag{2-8}$$

Substituting the estimate of A in Eq.(2-5) yields to

$$J(\theta) = \sum_{i=0}^{N-1} s_i^{\,2} - \frac{\left(\sum_{i=0}^{N-1} s_i R_i(\theta-\theta_i)\right)^2}{\sum_{i=0}^{N-1} R_i^2(\theta-\theta_i)} \tag{2-9}$$

Since main concern is the minimization of the Eq.(**2-9**) with respect to θ, the first term, which is not a function of θ, can be eliminated. Rearranging the equation yields

$$M(\theta) = \frac{\left(\sum_{i=0}^{N-1} s_i R_i(\theta-\theta_i)\right)^2}{\sum_{i=0}^{N-1} R_i^2(\theta-\theta_i)} \tag{2-10}$$

The AOA can be estimated by the value that maximizes $M(\theta)$.

$$\theta = \arg\max_{\theta}\left(M(\theta)\right) \tag{2-11}$$

2.2 Phase Comparison Algorithm

The DF system considered in this study consists of a linear phase array of two antennas, which are assumed to have identical gain patterns. The array geometry of the antennas is given in Figure 2.2.

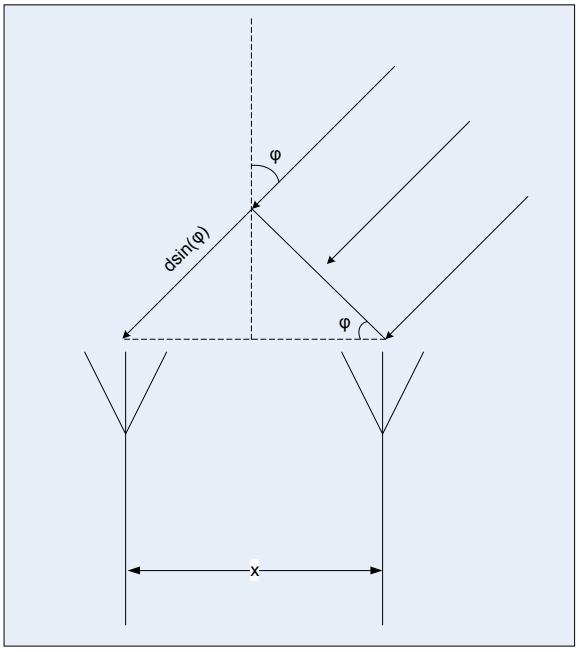

Figure 2.2 Phase Array Geometry

The signals arriving to antennas at angle of ϕ are given by

$$s = Ae^{j\theta} g(\phi) + n_1 \quad \textbf{(2-12)}$$

$$d = Ae^{j\theta} e^{j\psi} g(\phi) + n_2 \quad \textbf{(2-13)}$$

where, A is the amplitude and θ is the phase of the incoming signal, ψ is the phase difference between the signals arriving to the antennas, $g(\phi)$ is the antenna pattern and n_i is the receiver noise of the i^{th} antenna (i=1, 2).

In order to make Eq.(2-13) more compact, $h(\phi)$ is defined as

$$h(\phi) = e^{j\psi} g(\phi) \tag{2-14}$$

Substituting Eq.(2-14) in Eq.(2-12) and Eq.(2-13), received signals are redefined as

$$s = Ae^{j\theta} g(\phi) + n_1 \tag{2-15}$$

$$d = Ae^{j\theta} h(\phi) + n_2 \tag{2-16}$$

The additive noise n_i (i=1, 2) is assumed to be white Gaussian noise with noise samples being independent of each other and having identical variances σ^2. The Gaussian distributed random variable having zero mean and a variance of σ^2 has the density function given in Eq.(2-2). In this case, there are two independent random variables. Therefore their joint density function is the product of the densities for each Gaussian variable. The probability density function (pdf) is given by

$$f(A,\theta,\phi) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left|s - Ae^{j\theta} g(\phi)\right|^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left|d - Ae^{j\theta} h(\phi)\right|^2}{2\sigma^2}} \tag{2-17}$$

The principle is the same for all maximum likelihood estimation problems. The maximum likelihood estimate for parameter ϕ is the one which makes the given observations the most likely. In other words, it is the value that maximizes $f(A,\theta,\phi)$.

$$\phi = \arg\max_{\varphi} \left(f(A,\theta,\phi) \right) \tag{2-18}$$

Since logarithm is a monotonic function, maximizing the Eq. (2-17) is equivalent to minimizing the negative logarithm of the likelihood function. By eliminating constant terms of the function, the maximum likelihood function becomes

$$J(A,\theta,\phi) = \left| s - Ae^{j\theta} g(\phi) \right|^2 + \left| d - Ae^{j\theta} h(\phi) \right|^2 \tag{2-19}$$

Now the maximum likelihood problem can be estimated by the value that minimizes $J(A,\theta,\phi)$.

$$\phi = arg\min_{\phi}(J(A,\theta,\phi)) \tag{2-20}$$

As it is observed in Eq.(2-19), the maximum likelihood estimation function is a function of A, θ and ϕ. To estimate AOA, derivatives of the function must be taken with respect to A, θ and set to 0.

Taking derivate of $J(A,\theta,\phi)$ with respect to A and setting to 0 yields to

$$\frac{\partial J(A,\theta,\phi)}{\partial A} = 2A|g(\phi)|^2 + 2A|h(\phi)|^2 - 2\,\mathrm{Re}\{se^{-j\theta}g^*(\phi)\} - 2\,\mathrm{Re}\{de^{-j\theta}h^*(\phi)\} = 0 \tag{2-21}$$

$$\hat{A} = \frac{\mathrm{Re}\{\mathrm{s.e}^{-\mathrm{j}\theta}.\mathrm{g}^*(\phi)\} + \mathrm{Re}\{\mathrm{d.e}^{-\mathrm{j}\theta}.\mathrm{h}^*(\phi)\}}{|\,g(\phi)|^2 + |\,h(\phi)|^2} \tag{2-22}$$

Taking derivate of $J(A,\theta,\phi)$ with respect to θ and setting to 0 yields to

$$\frac{\partial J(A,\theta,\phi)}{\partial \theta} = -\frac{\partial}{\partial \theta}\left((\mathrm{se}^{-\mathrm{j}\theta}\mathrm{g}^*(\phi) + \mathrm{s}^*\mathrm{e}^{\mathrm{j}\theta}\mathrm{g}(\phi)\right) - \frac{\partial}{\partial \theta}\left((\mathrm{se}^{-\mathrm{j}\theta}\mathrm{h}^*(\phi) + \mathrm{d}^*\mathrm{e}^{\mathrm{j}\theta}\mathrm{h}(\phi)\right) = 0 \tag{2-23}$$

Simplifying and rearranging the terms yields to

$$\mathrm{Im}\{e^{-j\theta}(sg^*(\phi) + dh^*(\phi))\} = 0 \tag{2-24}$$

Since the imaginary part of the equation is zero, it can be rewritten as

$$sg^*(\phi) + dh^*(\phi) = Ke^{j\theta + jk\pi} \tag{2-25}$$

The estimated value of θ can be simplified as

$$\hat{\theta} = \arg(sg^*(\phi) + dh^*(\phi)) \tag{2-26}$$

The estimate of A can be further simplified by substituting Eq.(2-26) in Eq.(2-22) as

$$\hat{A} = \frac{\left|\mathrm{sg}^*(\phi) + \mathrm{dh}^*(\phi)\right|}{|\,g(\phi)|^2 + |\,h(\phi)|^2} \tag{2-27}$$

Substituting the estimate of A found in Eq.(2-27) in $J(A,\theta,\phi)$, rearranging and simplifying gives

$$J(A,\theta,\phi)=|s|^2+|d|^2-\frac{\left|\mathrm{sg}^*(\phi)+\mathrm{dh}^*(\phi)\right|^2}{|g(\phi)|^2+|h(\phi)|^2} \tag{2-28}$$

Since the magnitude of the complex signals are constant and constitute a DC value, the equation can be further simplified by eliminating the constant non-negative terms.

$$J'(\phi)=\frac{\left|\mathrm{sg}^*(\phi)+\mathrm{dh}^*(\phi)\right|^2}{|g(\phi)|^2+|h(\phi)|^2} \tag{2-29}$$

In this method, direction of arrival, ϕ, is estimated by the value which maximizes the maximum likelihood function $J'(\phi)$.

$$\phi=\arg\max_{\phi}\left(\frac{\left|\mathrm{sg}^*(\phi)+\mathrm{dh}^*(\phi)\right|^2}{|g(\phi)|^2+|h(\phi)|^2}\right) \tag{2-30}$$

CHAPTER III

IMPLEMENTATION

3.1 Amplitude Comparison

The amplitude comparison system implemented in this work consists of six antennas uniformly located around a circular axis to have azimuth coverage of 360 degrees. The angle between the beam axes of two adjacent antennas is 60 degrees. The antennas are assumed to have identical antenna gain patterns.

The DF system is designed to have six receiver channels following the antennas. The signals received by the antennas pass through RF sections and are sampled by Analog to Digital Converters (ADC). After the received signals are sampled by ADCs, they are directed to FPGA. It is assumed that there exists a Pulse Descriptor Word (PDW) Generator block, implemented on FPGA platform. The purpose of this block is to extract the properties of the received signal, like frequency, pulse amplitude, pulse width etc. Then pulse amplitudes of the received signal from each receiver channel are input to the amplitude comparison block. The block diagram of the system is given in Figure 3.1.

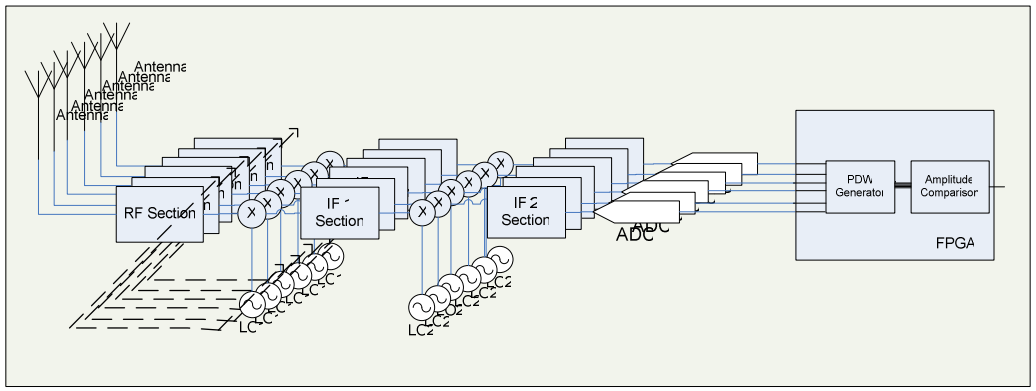

Figure 3.1 Block Diagram of the Amplitude Comparison DF System

The antenna plays a very important role in reception of the signals from the emitters. In this part of the study, ideal antenna gain pattern is generated and utilized. The specifications of the generated antenna pattern are given in Appendix A.

Throughout the experiments single operating frequency is considered. However, gain pattern of the antennas are not identical in practical DF systems. Moreover, the gain pattern of the antennas changes as the operating frequency changes. In other words, each antenna has distinct characteristics at varying operating frequencies, and distortion of the main beam axis direction and beamwidth are usually encountered. In this work, the errors due to different operating frequencies and pattern mismatches are not handled.

The generated antenna pattern is quantized at 1 degree resolution. The gain pattern of the antennas is given in Figure 3.2.

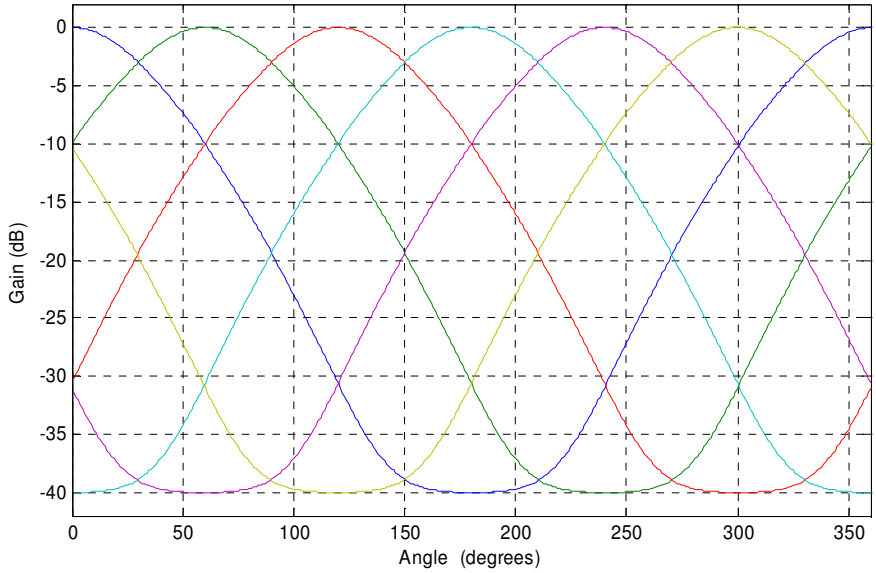

Figure 3.2 Amplitude Comparison Antenna Gain vs. Angle

3.1.1 Software Simulation Results

The amplitude comparison method, which is formulated in Chapter II, is simulated to obtain the DF accuracy of the algorithm taking into consideration different SNR values. Since the antenna pattern is quantized at 1 degree resolution, the estimation of emitter angular location has 1 degree resolution. Therefore the received signals are simulated as if emitter is angularly located in steps of 1 degree with respect to the DF system. In the software simulations, complex receiver noise is generated as if the signal is received by an isotropic antenna, which has a gain of 0dB. For each channel at specified SNR value, complex noise is generated and is added to the received signal under consideration. Two approaches are proposed for applying the amplitude comparison algorithm.

3.1.1.1 1st Approach

In this approach, the signals received from each channel in the system are directly used in the algorithm for locating the angular position of emitter.

There exist six identical regions in the 360 degree azimuthal coverage, since the gain patterns of the antennas are identical. Therefore AOA interval is selected between the intersections of gain patterns of the adjacent antennas. The antenna having a boresight of 60 degrees is taken as a reference. Search Interval is set to between 30 and 90 degrees. For each corresponding AOA, $3x10^4$ trials are performed. SNR value is varied from 10dB to 40dB in steps of 5dB. Simulations results are given in Figure 3.3, Figure 3.4 and Figure 3.5 respectively.

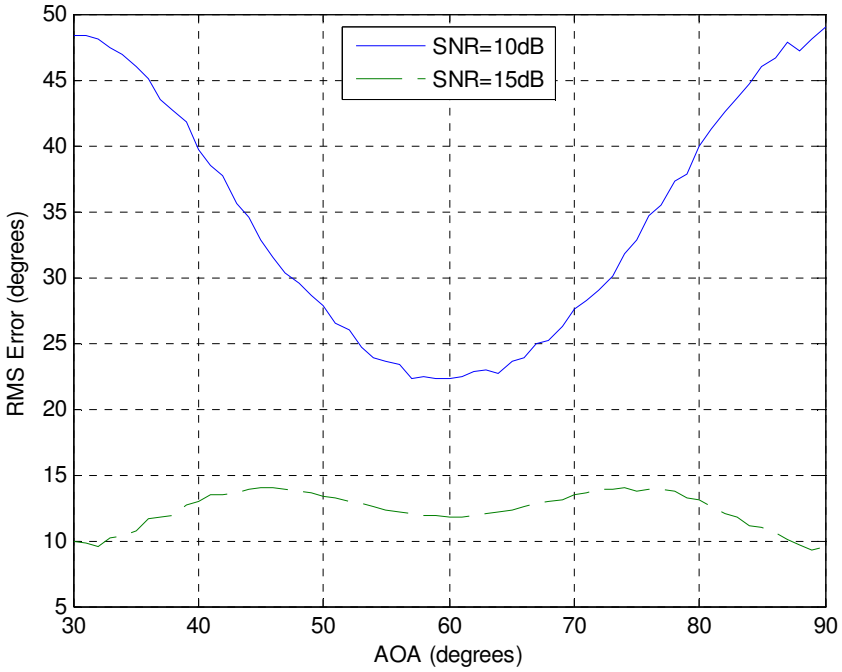

Figure 3.3 RMS Error vs. AOA (SNR is equal to 10dB and 15dB)

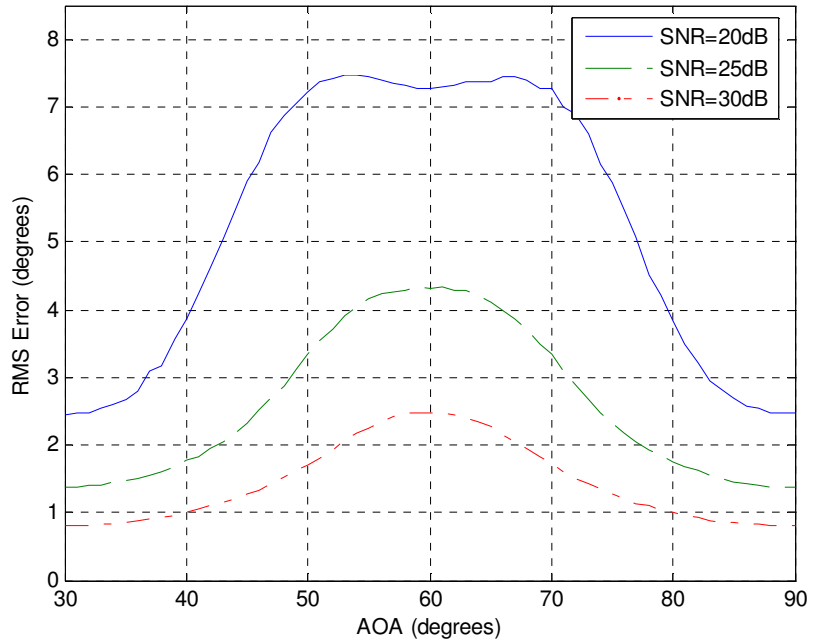

Figure 3.4 RMS Error vs. AOA (SNR is equal to 20dB, 25dB and 30dB)

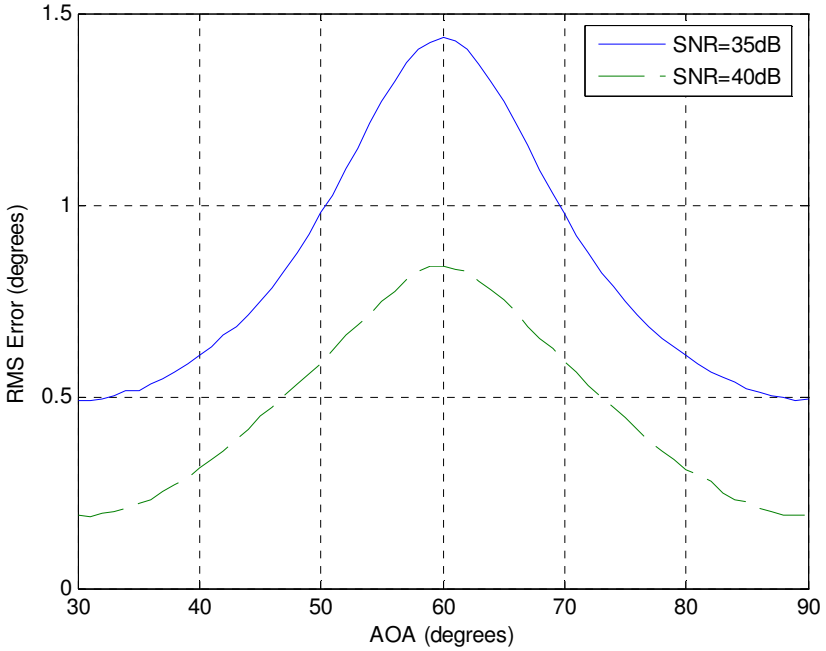

Figure 3.5 RMS Error vs. AOA (SNR is equal to 35dB and 40dB)

The DF accuracies with varying SNR values are summarized in Table 3-1, where statistical indicators, mean and the standard deviation of the RMS error are given.

Table 3-1 Mean and Standard Deviation of RMS Error (1st Approach)

	Mean (degrees)	Standard deviation (degrees)
RMS Error (SNR=10dB)	34.395	9.344
RMS Error (SNR=15dB)	12.330	1.3741
RMS Error (SNR=20dB)	5.273	2.013
RMS Error (SNR=25dB)	2.623	1.105
RMS Error (SNR=30dB)	1.448	0.591
RMS Error (SNR=35dB)	0.841	0.321
RMS Error (SNR=40dB)	0.467	0.222

3.1.1.2 2nd Approach

In this approach the pulse amplitudes from three antennas, which are directed to the emitter, are used instead of using all of the received pulse amplitudes. The signals received by remaining three antennas are eliminated, since they are mainly receiving the receiver noise. The antenna that receives the maximum pulse amplitude is determined. Then pulse amplitudes received from the antennas, which are adjacent to this antenna, are selected. These three pulse amplitudes are taken into consideration to be used in the DF algorithm. The corresponding RMS Error curves for different SNR values from 10dB to 40dB in steps of 5dB are plotted in Figure 3.6, Figure 3.7 and Figure 3.8.

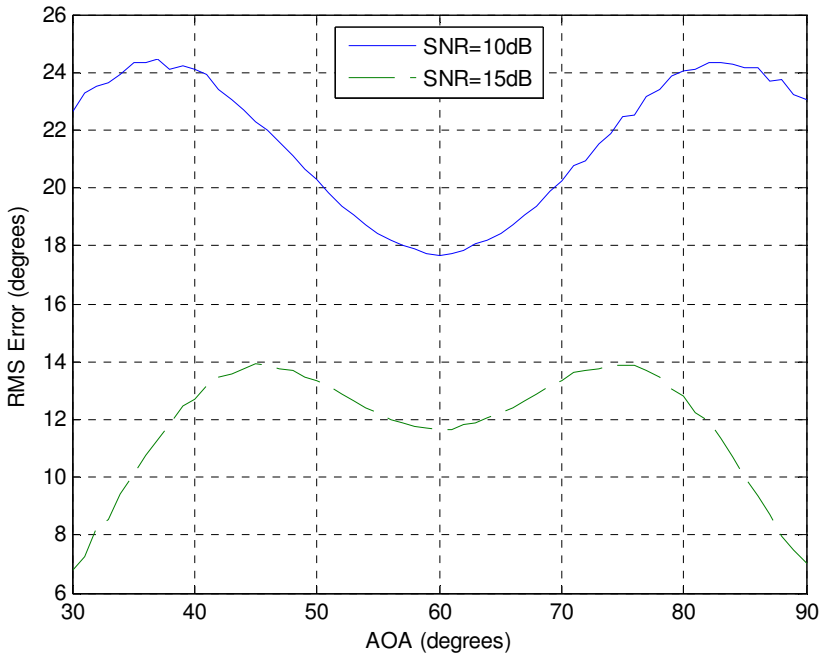

Figure 3.6 RMS Error vs. AOA (SNR is equal to 10dB and 15dB)

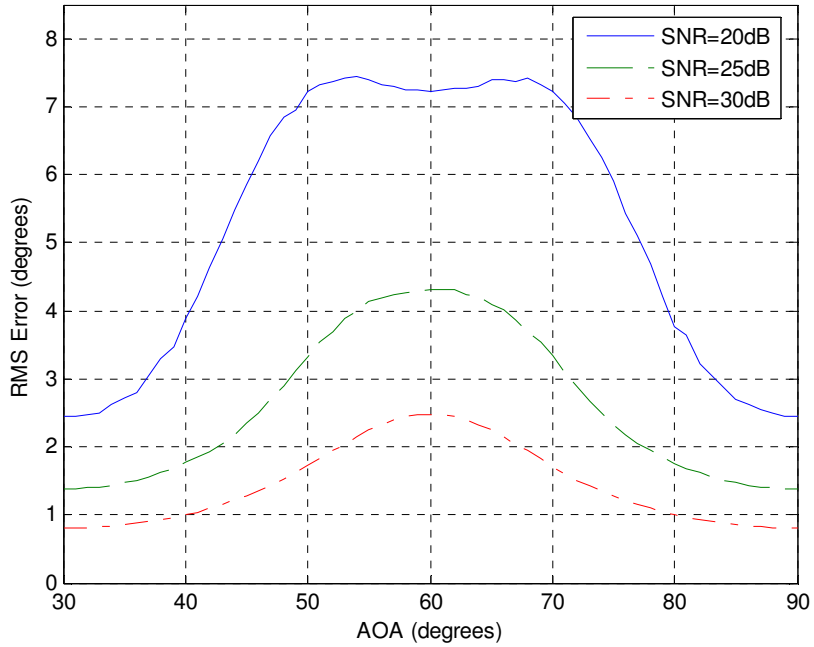

Figure 3.7 RMS Error vs. AOA (SNR is equal to 20dB, 25dB and 30dB)

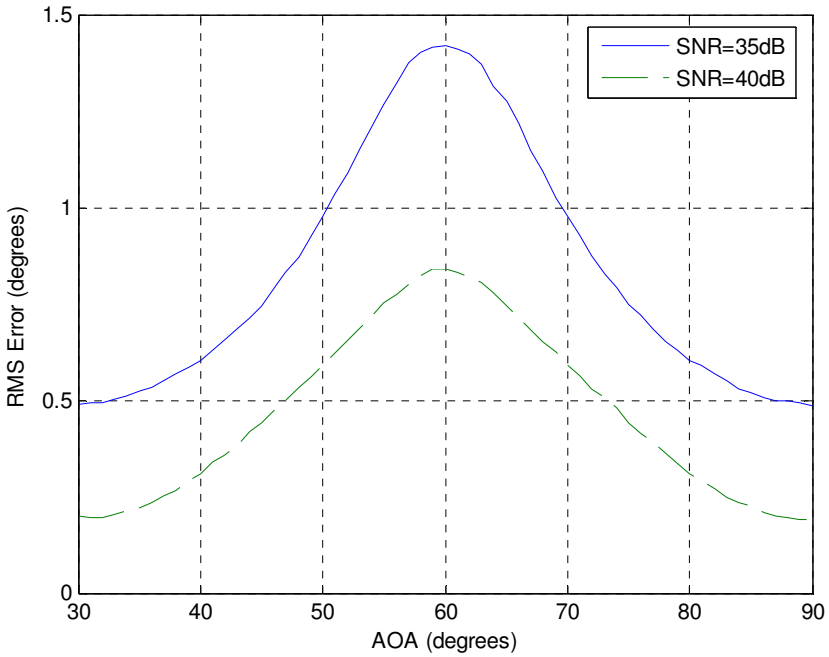

Figure 3.8 RMS Error vs. AOA (SNR is equal to 35 dB and 40dB)

Mean and the standard deviation of the RMS Error are calculated for different SNR values and are given in Table 3-2.

Table 3-2 Mean and Standard Deviation of RMS Error (2nd Approach)

	Mean (degrees)	Standard deviation (degrees)
RMS Error (SNR=10dB)	21.636	2.377
RMS Error (SNR=15dB)	11.879	1.964
RMS Error (SNR=20dB)	5.222	1.99
RMS Error (SNR=25dB)	2.62	1.097
RMS Error (SNR=30dB)	1.446	0.588
RMS Error (SNR=35dB)	0.840	0.319
RMS Error (SNR=40dB)	0.467	0.221

According to simulation results, even at the same SNR, the RMS error alters along DOA of the source. It is observed that, for the low SNR cases AOA estimate of the sources, which are located in the vicinity of direction of the antenna boresight, are more accurate. The main reason is that the antennas receive high levels of noise which prevents gathering proper measurement of the pulse amplitude. On the other hand as SNR increases, AOA estimates of the sources, located in the middle of the antenna boresights, become more accurate. The reason for this is that SNR is high and contribution of the other antennas to algorithm makes the DF accuracy better. At the boresight amplitude fluctuations due to the presence of noise affect the DF accuracy more, because the antenna gain pattern is linearly stored and is very flat around the boresight. Therefore, within the beamwidth of the antenna, as AOA gets further from the boresight, the slope of the antenna gain increases. Consequently better DF accuracy is obtained.

3.1.1.3 Comparison of Approaches

When two approaches are compared, 2nd approach, which uses pulse amplitudes from 3 antennas, is superior to 1st approach, in which pulse amplitudes from all receiving channels are used. For low SNR cases DF accuracy of the 2nd approach is better than the 1st approach. However as SNR value increases both algorithms have approximately the same DF accuracy. However, considering the overall performance, the 2nd approach is more preferable; therefore 2nd approach is selected for the hardware implementation.

Mean of the RMS error is plotted in Figure 3.9 for both approaches.

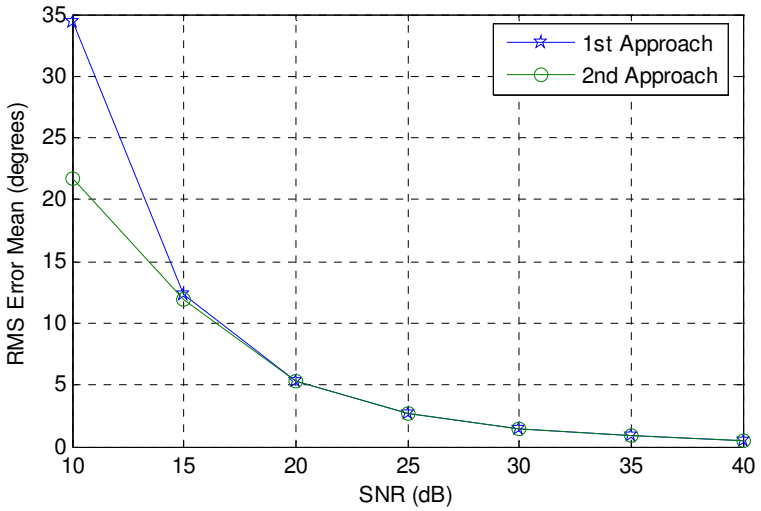

Figure 3.9 Mean of RMS Error vs. SNR

3.1.2 Hardware Implementation

In this part, implementation of the amplitude comparison DF algorithm is implemented on an FPGA platform by using a commercial software tool, Xilinx System Generator Tool, is discussed. General information about the software tool and development tool flow is given in Appendix B.

In the software simulations, floating point double precision values were used to determine angular location. However, when the algorithm is implemented on FPGA, fixed point arithmetic is used. As a result the errors due to quantization are inevitable.

Here the question is how many bits are required for each signal to be represented in the FPGA. As a general rule for hardware implementation, the number of bits used in the design is directly proportional to the resources used in FPGA. Therefore the width of the data should be determined carefully. In the design, actually there are two signals that data width should be considered carefully; first one is the pulse amplitude, the other is the antenna gain pattern, which is stored in the RAM.

After the signals are received by antennas, they pass through the RF sections and before processed in FPGA fabric, the IF signals are converted from analog to digital by ADCs. In today's technology, throughput of commercial ADC varies from few MSPS to 12 GSPS [1]. Generally as throughput of the ADC increases, the number of bits that represent resolution decreases. In today's technology, ADCs whose throughput rate is greater than 200 MSPS mostly have 8, 10 or 12 bits for resolution. Therefore, considering this fact and the process gains in FPGA, it is a good approximation to determine 12 bits for the pulse amplitude.

The other parameter, whose data width should be determined, is the antenna gain pattern. Since data width of the received signal pulse amplitude is selected to be 12 bits, the antenna pattern data width should be comparable to data width of pulse amplitude. Large data widths of antenna gain pattern do not provide much resolution, since pulse amplitude is already limited to the 12 bits. Consequently, experiments were performed with 12 bit and 16 bit quantized antenna gain patterns, regarding different SNR cases.

3.1.2.1 Hardware Design

In the hardware design, inputs to the algorithm are pulse amplitude information from each receiver channel and a strobe, indicating that the pulse amplitudes are valid and ready for processing. The output of the hardware implementation is the AOA estimate of the emitter. Operating frequency is assumed to be constant.

The hardware design is composed of three main blocks. These blocks are;

- AC_Angle_Counter, which filters the necessary pulse amplitudes and generates a valid search interval.
- AC_Correlation_Computation, which calculates the ML function.
- AC_AOA_Estimator, which estimates the angular location of the emitter.

The purpose and the operational details of each of the implemented block are presented in the following parts of this chapter. The block diagram of the hardware implementation is given in Figure 3.10.

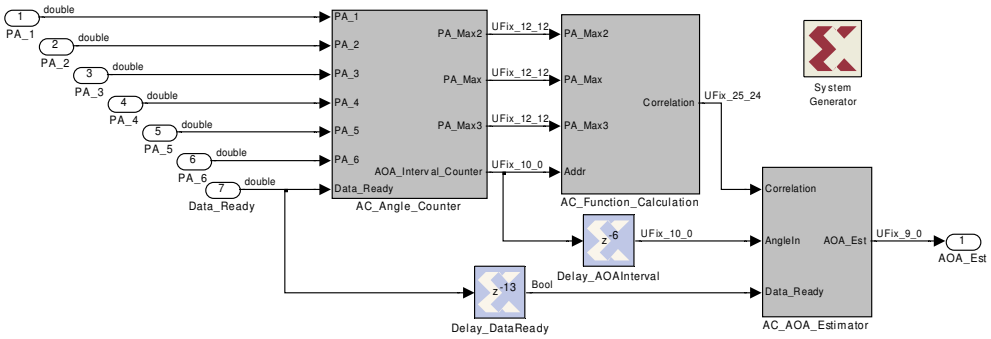

Figure 3.10 Block Diagram of the Amplitude Comparison Implementation

3.1.2.1.1 AC_Angle_Counter Block

The purpose of this block is to generate a search interval for estimation of AOA. In monopulse DF systems, not only the DF accuracy but also the process time needed to determine the AOA is very important. Since the speed of the algorithm is very crucial, it is better to make calculations in a smaller interval than making calculation for the entire coverage range.

Implementation of the Angle_Counter consists of three hierarchical blocks.

- AC_Max_PA_Index_Finder, which determines the index of the Antenna receving maximum pulse amplitude.
- AC_Max_PA_Selector, which selects the proper pulse amplitudes.
- AC_Address_Counter, which generates a valid search interval.

Logic diagram of the *AC_Angle_Counter* block is given in Figure 3.11.

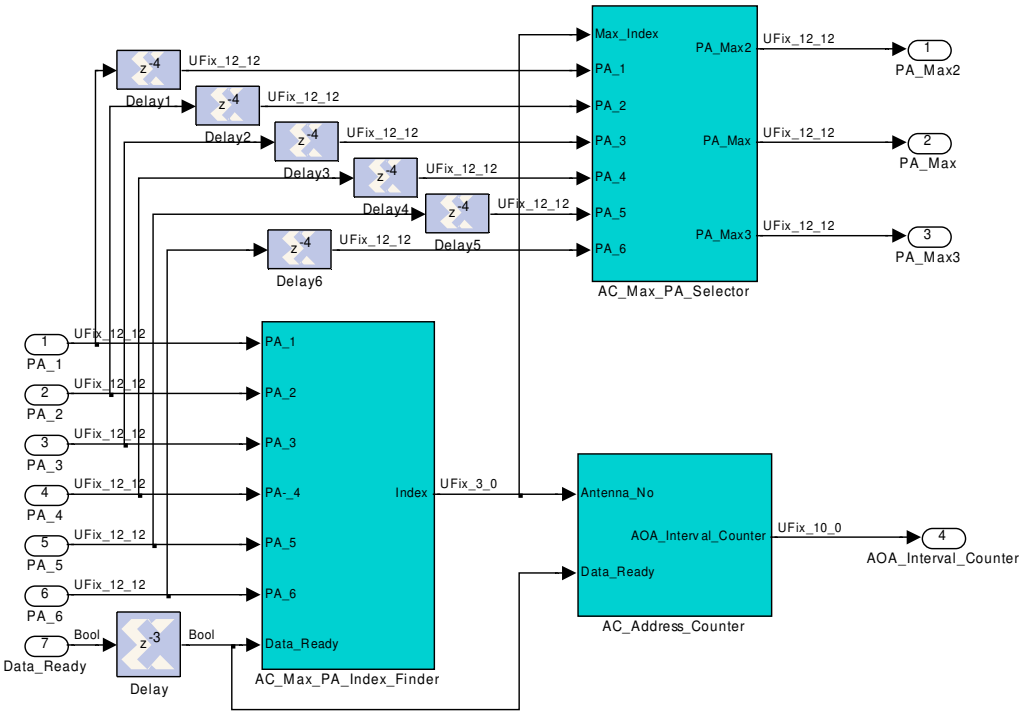

Figure 3.11 Logic Diagram of AC_Angle_Counter Block

3.1.2.1.1.1 AC_Max_PA_Index_Finder Block

The purpose of this block is to determine the antenna, at which the maximum pulse amplitude is received. Pulse amplitudes of the received signals are input to this block and output is the antenna index.

The logic diagram of the *AC_Max_Amplitude_Finder* block is given in Figure 3.12.

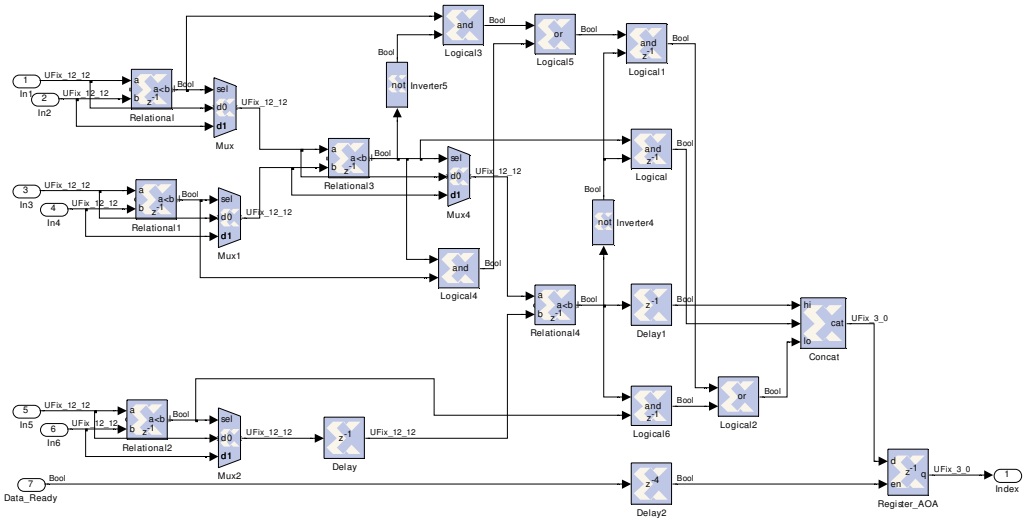

Figure 3.12 Logic Diagram of AC_Max_Amplitude_Finder Block

3.1.2.1.1.2 AC_Max_PA_Selector Block

The block is designed to determine the maximum pulse amplitude and pulse amplitudes from antennas which are adjacent to the antenna receiving the maximum pulse amplitude. The inputs are pulse amplitudes from each receiver channel and the index of the antenna which received the maximum pulse amplitude. Depending on the *Max_Index* input, three pulse amplitudes are filtered. Logic diagram of the *AC_Max_PA_Selector* block is given in Figure 3.13.

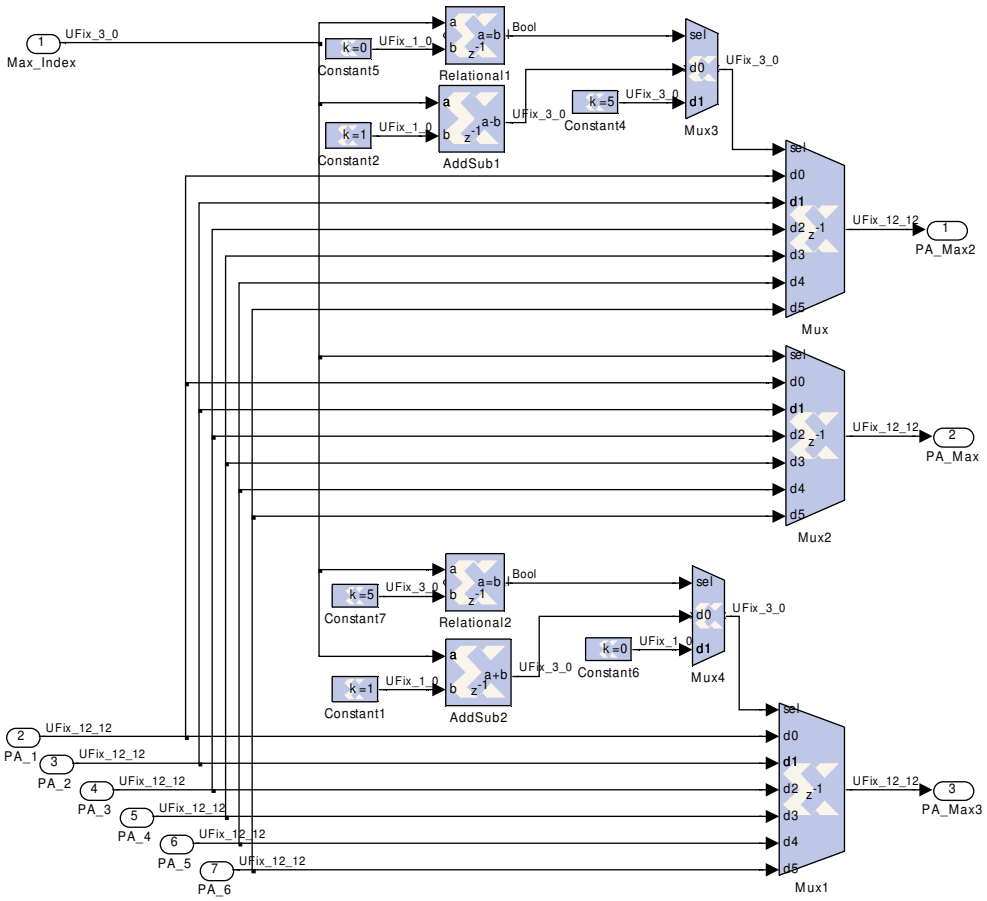

Figure 3.13 Logic Diagram of AC_Max_PA_Selector Block

3.1.2.1.1.3 AC_Address_Counter Block

The purpose of this block is to generate a sequential search interval for estimating AOA depending on the antenna, which receives the signal at maximum amplitude. The determined search interval is 60 degrees around the boresight of this antenna.

The logic diagram of the *AC_Address_Counter* block is given in Figure 3.14.

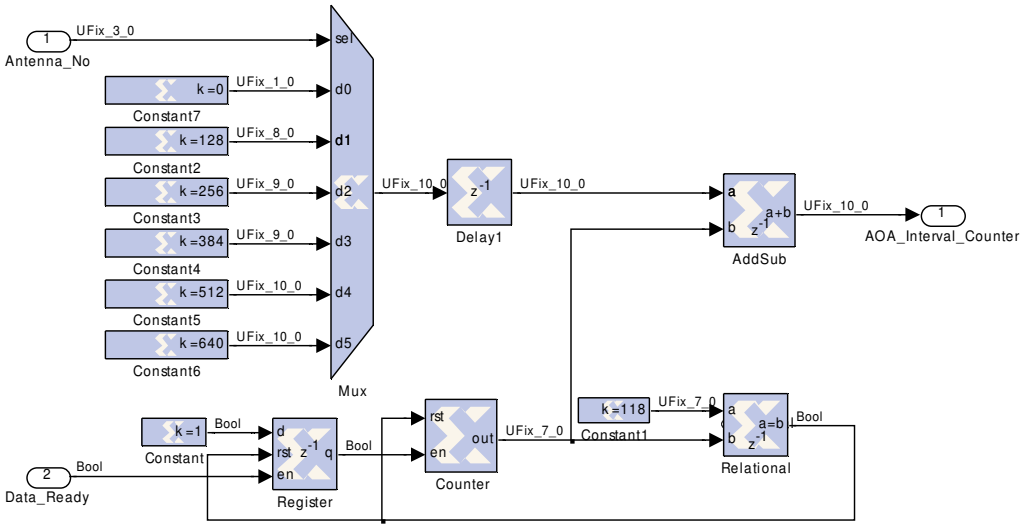

Figure 3.14 Logic Diagram of AC_Address_Counter Block

When *Data_Ready* strobe is high, the counter value is reset. According to the *Antenna_No* input, the beginning of the AOA interval is determined. Then counter counts up to 120, which is the predetermined search interval. Taking into account the low SNR cases, this interval should be greater than 60 degrees, 3dB beamwidth. The center of the search interval depends on the antenna, which receives the greatest pulse amplitude. The possible search intervals are given in Figure 3.15.

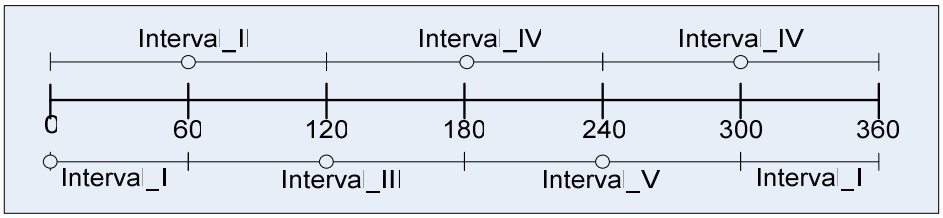

Figure 3.15 Search Interval for Amplitude Comparison Implementation

For the high SNR cases, alternatively the search interval can be decreased up to 3 dB beamwidth of the antenna. This change will result in reduced computation time, which yields an improvement of total process time for the estimation of AOA.

3.1.2.1.2 AC_Correlation_Computation Block

In order to find the AOA of the incoming signal, as stated in Chapter II, the algorithm tries to find the argument that maximizes $M(\theta)$ in Eq. (2.10). This block is implemented for calculating $M(\theta)$. It will be useful to rewrite the expression for $M(\theta)$ here.

$$M(\theta)=\frac{\left(\sum_{i=0}^{N-1} s_i R_i(\theta-\theta_i)\right)^2}{\sum_{i=0}^{N-1} R_i^2(\theta-\theta_i)} \tag{2.10}$$

The logic diagram of the block is given in Figure 3.16.

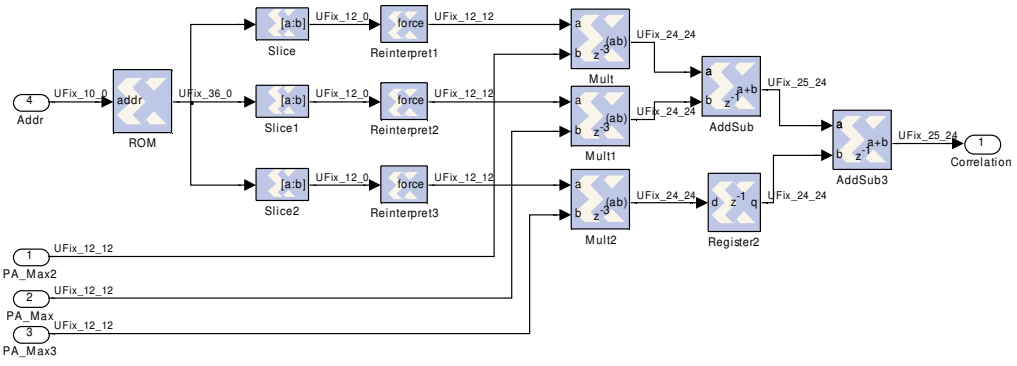

Figure 3.16 Logic Diagram of the AC_Correlation_Computation Block

AC_Correlation_Computation block inputs three pulse amplitudes and the AOA search interval from previous block. The search interval, *Adr*, is used for addressing the ROM block and taking corresponding antenna gain value at the specified angle.

At the hardware implementation phase, a modification is made on the above equation in order to simplify and eliminate the operation complexity in the hardware. Taking the square root of both the denominator and the numerator does not affect the estimated AOA value, since s_i and R_i are non-negative real values. Moreover, simplifications will be quite useful for storing the modified antenna gain pattern. Taking the square root of the Eq. (2-10) yields

$$M'(\theta) = \frac{\sum_{i=0}^{N-1} s_i R_i(\theta - \theta_i)}{\sqrt{\sum_{i=0}^{N-1} R_i^2(\theta - \theta_i)}} \quad \textbf{(3-1)}$$

Antenna pattern is generated at each angle from 1 to 360 degrees at a resolution of 1 degree. At every angle, antenna pattern is normalized and stored. Normalization is performed according to the following equation.

$$R_i'(\theta) = \frac{R_i(\theta - \theta_i)}{\sqrt{\sum_{i=0}^{N-1} R_i^2(\theta - \theta_i)}} \quad \textbf{(3-2)}$$

Each memory row in table contains three antenna gains at the same angle. While forming antenna gain pattern tables, gains are quantized at n bits and merged into 3n bits wide. Therefore after addressing ROM block, the value received is sliced into three parts and multiplied with the pulse amplitude values. To calculate $M'(\theta)$ at the specified angle, the products are summed up and released as an output of the block.

The overall pipelined latency of this block is 6 clock cycles for each angle value. Reading table data from the memory is performed in a single clock cycle. In order to minimize the process time, multiplication and addition of the values are done in parallel in 5 clock cycles. As a result, the value of the function is ready at the output of the block for each AOA, after a process delay 6 clock cycles.

3.1.2.1.3 AC_AOA_Estimator Block

This block is the final block that estimates the AOA of the incoming signal. The inputs to this block are the angle within the search interval, value of $M'(\theta)$ at this angle and the strobe, indicating that the pulse amplitudes are ready. Output of this block is the estimated AOA value.

The logic diagram of the block is given in Figure 3.17.

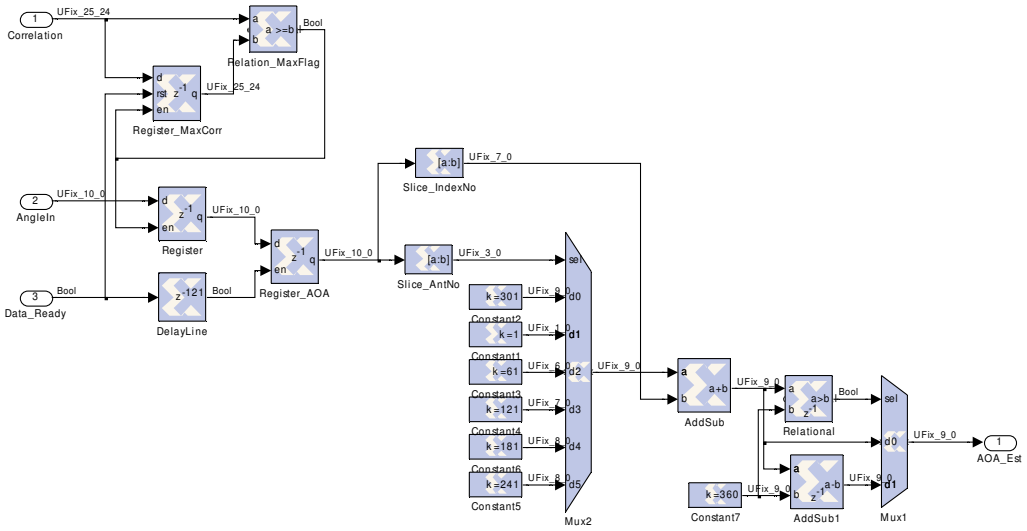

Figure 3.17 Logic Diagram of AC_AOA_Estimator Block

The function of this block is to find the angle, at which maximum of the function occurs, over the predefined search interval. When a maximum point is found, angle value corresponding to this point is latched, and at the end of search interval, this value is presented as an output of the block.

In order to synchronize the *Correlation*, *Angle_In* and *Data_Ready* signals and to make proper AOA estimate, the signals are delayed by using delay blocks in previous blocks.

After *Data_Ready* strobe makes a transition to high state, 137 clock cycles later *AOA_Est* is ready. The latency is as follows; 7 clock cycles in *AC_Angle_Counter* block to determine the search interval, 6 clock cycles to calculate the function $M'(\theta)$ 120 clock cycles to calculate the ML function and 4 clock cycles to find angle, at which the maximum of the ML function is encountered.

3.1.2.2 Simulation Results

As stated in the hardware implementation part, experiments were performed with 12 bit and 16 bit quantized antenna gain patterns, regarding different SNR cases. ($5x10^3$ Iterations were performed for each AOA value.)

In the hardware design, when performing a multiplication, adjustments are made in order not to lose the precision at the output. For the 12 bit simulation, the multiplication of two values, 12 bit unsigned where binary point 12^{th} bit and 16 bit unsigned where binary point 16^{th} bit, the product is 28 bit unsigned where binary point in 28^{th} bit. Similar procedure is also applied for 12 bit simulation. In addition, saturation arithmetic and rounding have area and performance costs [12]. Therefore they are used only if necessary.

The clock rate of the system is set to 200 MHz, which corresponds to 5 ns of clock period. While implementing multiplier blocks in FPGA, it is advantageous to use 18x18 dedicated embedded multipliers [11]. This provides flexibility in hardware design. Since these multipliers are dedicated, no FPGA resources are used. Therefore the logic, which is planned to be used for multiplication block, can be used for implementation of other blocks.

The screenshot of the scope from the hardware implementation is given in Figure 3.18. In the simulation, the pulses are generated with pulse repetition interval (PRI) of 1 μs., where SNR value is set to 30dB and AOA of the emitter is selected to be 250 degrees. In the Figure 3.18, search interval, $M'(\theta)$ function, Estimated AOA and *Data_Ready* signals are plotted for each pulse.

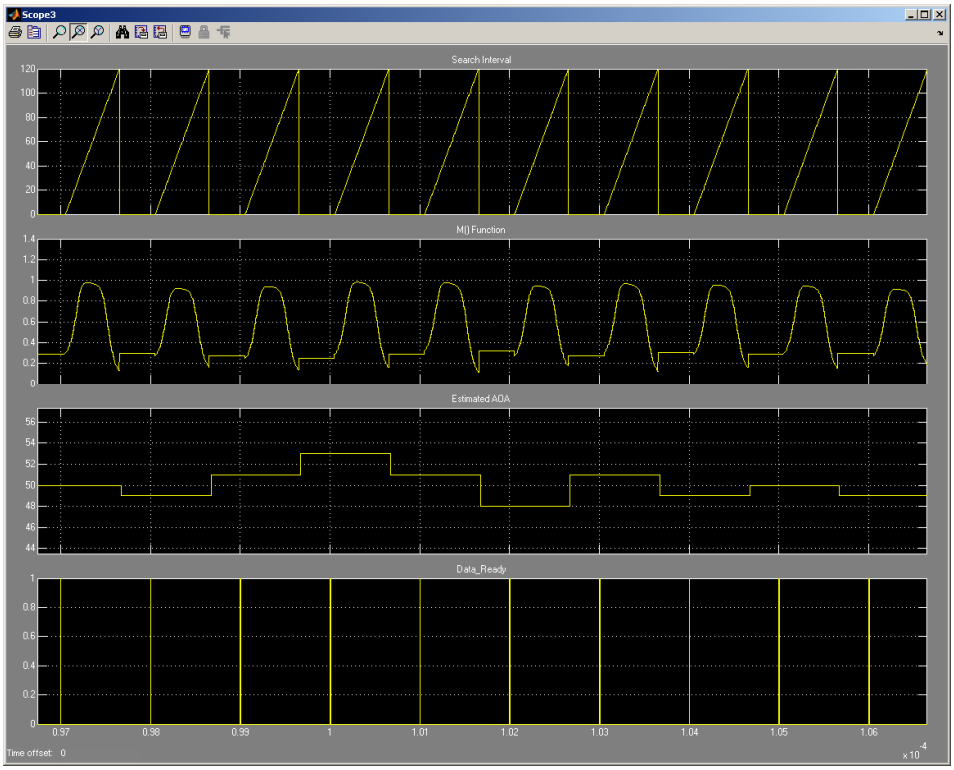

Figure 3.18 Screen Shot from System Generator (Implementation of Amplitude Comparison)

Since the antenna patterns are symmetric along the direction of the beam axis, there exist 6 identical zones in the 360 degree azimuth coverage that have the same antenna gain pattern. Therefore the experiments are performed in 60 degree interval at a resolution of 1 degree. RMS Error of the 12 bit and 16 bit hardware implementation simulations for 20dB SNR and 30 dB SNR are given in Figure 3.19 and Figure 3.20 respectively.

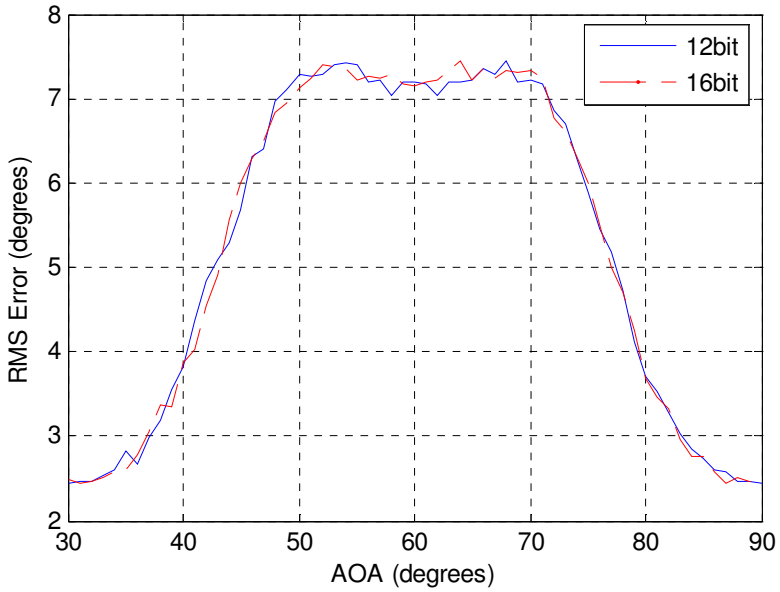

Figure 3.19 RMS Error vs AOA of Amplitude Comparison Hardware Implementation at 20dB SNR

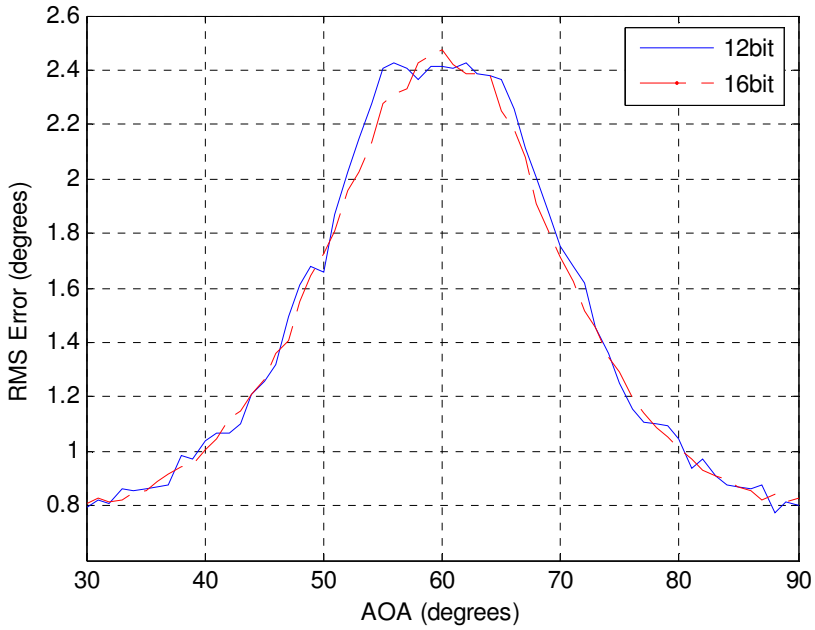

Figure 3.20 RMS Error vs AOA of Amplitude Comparison Hardware Implementation at 30dB SNR

Mean and standard deviation statistics of RMS Error of the 12 bit and 16 bit simulation results at 20dB and 30dB SNR are given in Table 3-3.

Table 3-3 Mean and Standard Deviation of RMS Error for Amplitude Comparison Hardware Implementation

	Mean (degrees)	Standard deviation (degrees)
RMS Error 12 bit (SNR = 20dB)	5.245	1,976
RMS Error 16 bit (SNR = 20dB)	5.240	1.994
RMS Error 12 bit (SNR = 30dB)	1.467	0.6
RMS Error 16 bit (SNR = 30dB)	1.448	0.58

Table 3-3 summarizes the performance of two hardware implementations having different number of bits for representing the antenna gain pattern. According to simulation results, storing the antenna gain pattern table 12 bit precision or 16 bit precision does not affect the DF accuracy of the algorithm much, since the amplitude of the signal is limited by 12 bits. But it is observed that In 16 bit implementation mean of estimated AOA values are more accurate due to greater quantization steps.

When these results are compared to software simulation results, both implementation results are very close to the results obtained in software simulation. Both 12 bits and 16 bits hardware implementations provide sufficient resolution, since calculated quantization errors are very small.

For the hardware implementation, the utilization of the FPGA resources is given in Table 3-4.

Table 3-4 FPGA Resource Utilization Summary of Amplitude Comparison Implementation

Number of Slice	396
Number of Slice Flip Flops	467
Number of 4 input LUTs	517
Number of IOBs	83
Number of MULT18X18s	3
Total equivalent gate count for design:	156.176

It is useful to give some information about the timing constraints. According to the generated clock report of the implementation, following results are obtained.

Requested Constraint : 5.000ns

Actual Constraint : 4.813ns

The number of signals not completely routed for this design is: 0 ns

The average connection delay for this design is 0.827 ns

The maximum pin delay is: 3.029 ns

The average connection delay on the 10 worst nets is: 2.338 ns

Listing Pin Delays by value: (ns)

d < 1.00 < d < 2.00 < d < 3.00 < d < 4.00 < d < 5.00 d >= 5.00

1823 850 61 1 0 0

According to synthesis report all constraints were met and all signals are completely routed.

3.2 Phase Comparison

The phase comparison system has two receiver channels following the antennas. Similar to the amplitude comparison system, the signals received by the antennas pass through RF paths and are sampled by ADC. After these phases, the digital signals are directed to FPGA. In the FPGA, it is assumed that Pulse Descriptor Word (PDW) Generator block measures parameters of the received signals in FPGA and extract the properties like pulse amplitude, frequency, pulse width etc. Part of this PDW Generator block supplies pulse amplitude and the frequency information of the received signal to phase comparison block. The block diagram of the system is given in Figure 3.21.

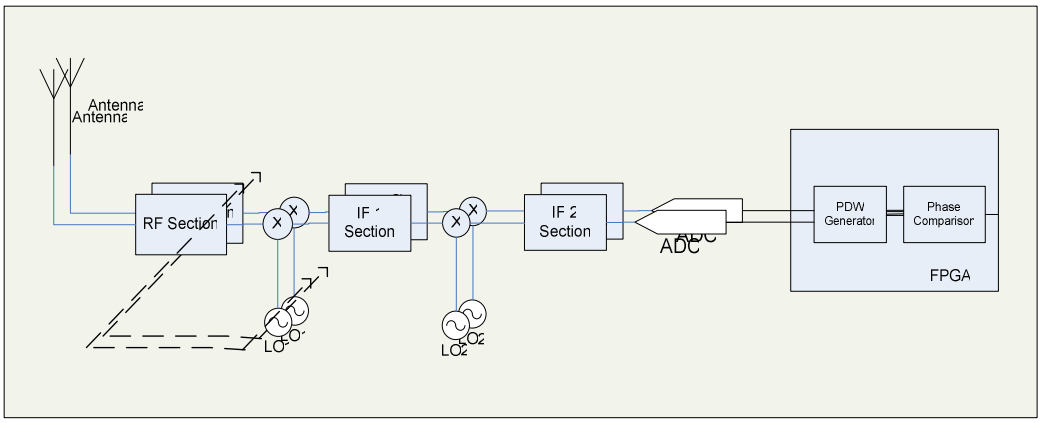

Figure 3.21 Block diagram of the Phase Comparison DF system

A single baseline interferometer is assumed in the phase comparison system. As shown in Figure 3.22, two antennas are located on a straight line with spacing of d. The antennas are assumed to have the same antenna gain patterns. The phase difference is measured by two identical antennas in space, separated by a finite baseline. The phase difference occurs from the extra time it took a plane wave to travel a greater distance to the further antenna. If the source of the signal is located at equal distance from the antennas, the phase difference would be zero. In this case, a phase difference of null arises at the boresight of the antenna.

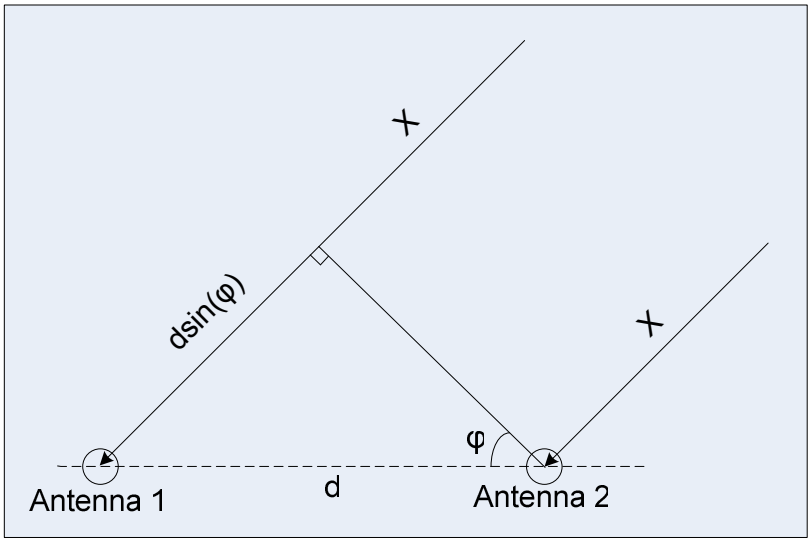

Figure 3.22 Single Baseline Interferometer

The signal received by Antenna 2 can be expressed as

$$S_2 = Ae^{jwt-\frac{2\Pi}{\lambda}X} \tag{3-3}$$

where, A represents initial signal amplitude, X is the distance traveled by the signal to reach the Antenna 2, λ is the wavelength of the traveling antenna and d is the distance between the antennas.

In the same manner, signal received by Antenna 1 is given by

$$S_1 = Ae^{jwt-\frac{2\pi}{\lambda}(X+d\sin\varphi)} \tag{3-4}$$

Where, $d\sin\varphi$ term represents the additional path to Antenna 1 as reference to Antenna 2. Since the important parameter in this case is phase difference of the arriving signals, $(2\pi/\lambda)X$ can be assumed to be zero. Subtracting the natural logarithm of the Eq.(3-4) from the natural logarithm of the Eq.(3-3) yields to

$$\ln S_2 - \ln S_1 = j\frac{2\pi}{\lambda}d\sin\varphi \tag{3-5}$$

Let θ be defined as a phase difference. Therefore θ can be expressed as

$$\theta = 2\pi \frac{d}{\lambda} \sin \varphi \qquad \textbf{(3-6)}$$

As an examination of the above equation, the value of d/λ ratio is very crucial. When d/λ ratio is equal to 1/2, the phase difference θ is in the interval between 0 and π. According to this interval, AOA φ has two solutions, which reside in the first quadrant and in the second quadrant. In the same manner, if d/λ is chosen to be 1, the phase difference is between 0 and 2π, and like in the previous case, there are two solutions, since phase difference is not a multiple of 2π. When d/λ ratio is selected to be n, where n is an integer greater than 1, the phase difference is in the interval between 0 and 2πn. In this case there are 2n solutions, n of which are in the first quadrant and the other n solutions reside in the second quadrant.

In the experimental work, AOA of the signals propagating from first quadrant is assumed to be positive and AOA of the signals from the left side of boresight will be regarded as negative. Therefore, AOA will be determined in the interval between -90 and 90 degrees, covering the 180 degree range.

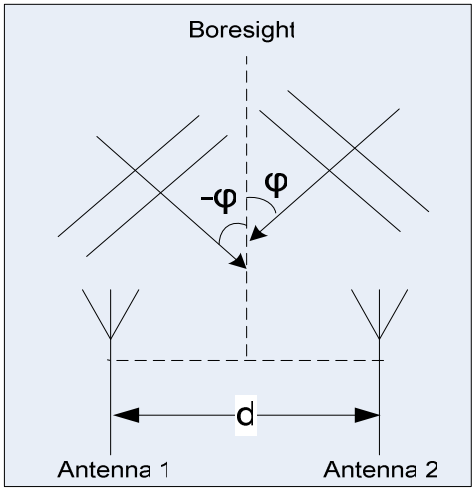

Figure 3.23 Phase Comparison Antenna Layout

As part of implementation of the phase comparison algorithm, software simulations are performed. In software simulations antennas are assumed to be omni-directional, since the phase difference of the received signals is the primary concern. According to the phase comparison algorithm stated in Chapter II, one of the antenna patterns is stored as complex valued. In Figure 3.24 real and imaginary parts and in Figure 3.25 phase and gain of the antenna pattern is plotted when d/λ is equal to 2.

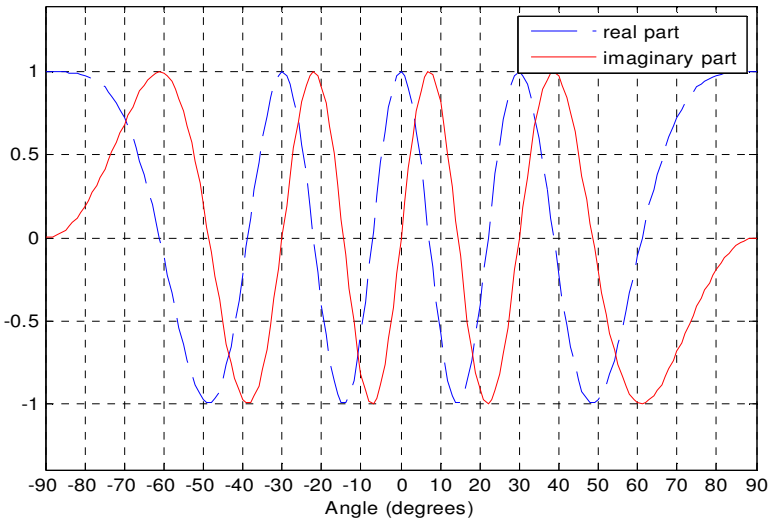

Figure 3.24 Real and Imaginary parts of Phase Comparison for d/λ=2

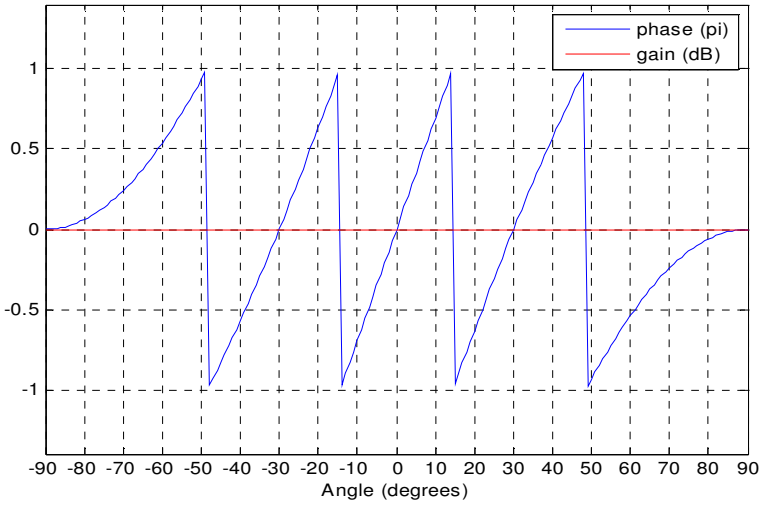

Figure 3.25 Phase Comparison Antenna Gain and Phase Pattern for d/λ=2

The AOA of the received signals can be estimated by the argument that maximizes the function $J(A,\theta,\phi)$. The typical plot of $J(A,\theta,\phi)$ is given in Figure 3.26, where direction of arrival is set to 35 degrees. The function is plotted for both when $d/\lambda = 1$ and $d/\lambda = 5$.

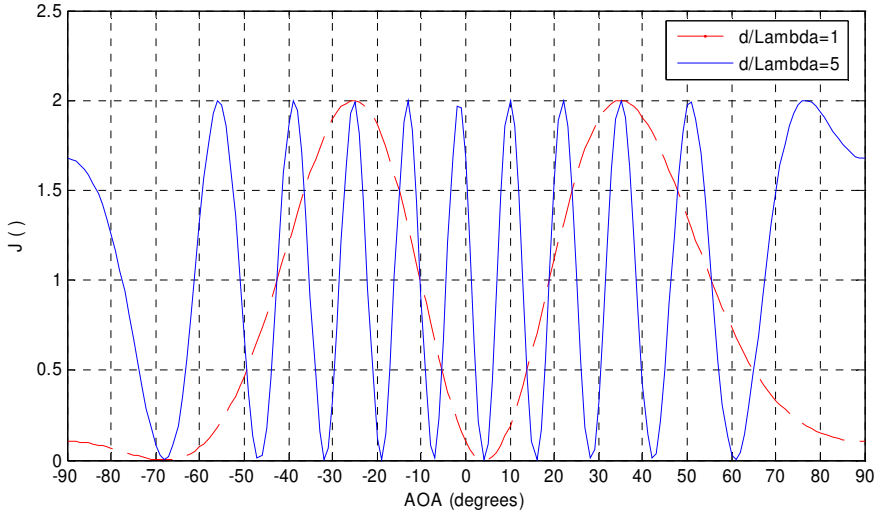

Figure 3.26 Typical Plot of $J'(\phi)$ vs. AOA for $d/\lambda = 1$ and $d/\lambda = 5$ (AOA=35)

The most important point of this function is that in each quadrant it has more than one local maximum point for d/λ values greater than 1. These points are possible AOA of the received signals.

3.2.1 Software Simulation Results

In order to get rid of the ambiguities and determine angular location, prior information of possible AOA should be present. As declared in [6], due to ambiguities, phase comparison systems work with another DF system to provide more accurate results. Therefore a search interval must be stated to find the maximum point of the function in order to determine the angular location.

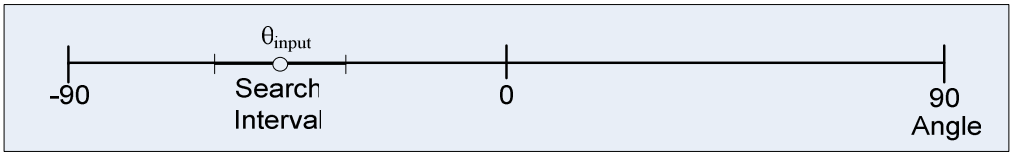

Figure 3.27 Search Interval for Phase Comparison

Therefore, a search interval of $\theta_{input} \pm \Delta\theta$ is defined, where θ_{input} represents the prior AOA information provided by another DF system. At the boundary point in the first quadrant, where $\theta_{input} + \Delta\theta$ is greater than 90 degrees, the interval limit is set to 90 degrees. At the second quadrant, where $\theta_{input} - \Delta\theta$ is smaller than -90, the interval limit is set to -90 degrees.

The factors that affect the choice of $\Delta\theta$ are d/λ ratio, the accuracy of phase comparison system under the defined SNR and the DF accuracy of the system that provides prior AOA estimate. Main objective is to detect only the maximum point of the lobe of interest over the search interval for accurate DF sensing. If great d/λ value and poor DF accuracy of the prior AOA information is considered, the determined search interval will not be appropriate. If the search interval is determined to be large, the other lobes, which create ambiguity, may exist in the search interval. On the other hand if the search interval is determined to be small,

the peak of the desired lobe may not be detected. Consequently in both cases it will lead to errors in the measurement of AOA which is an unwanted situation. Therefore to determine the search interval, the d/λ value and the DF accuracy of the other system should be considered and evaluated carefully where the DF accuracy of the phase comparison system should not be ignored.

To have a better understanding of d/λ ratio and the DF accuracy, some software simulations are performed. The results of the experiment varying d/λ ratio vs. RMS error plots at different SNR levels are given in Figure 3.28, Figure 3.29 and Figure 3.30.

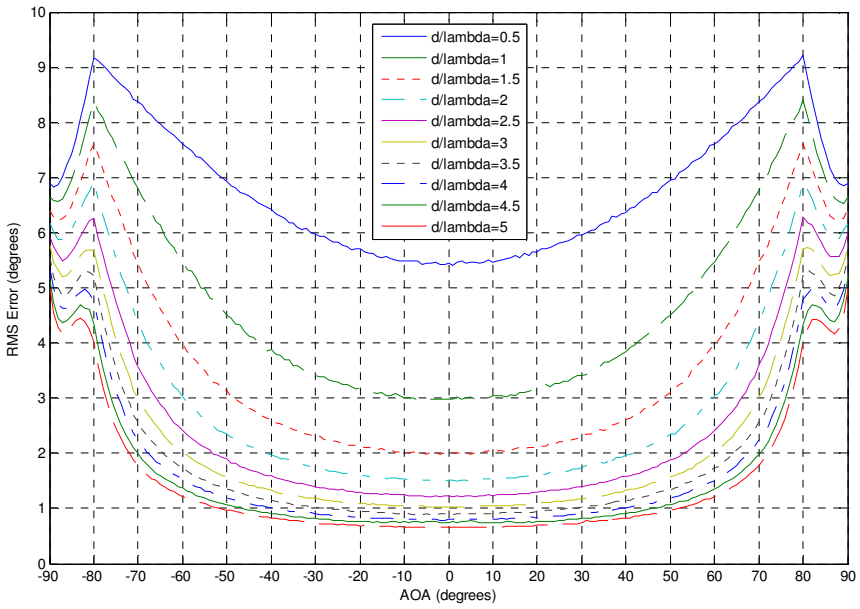

Figure 3.28 Phase Comparison RMS Error vs. AOA at d/λ value from 1 to 5 in steps of 0.5 (SNR = 10 dB)

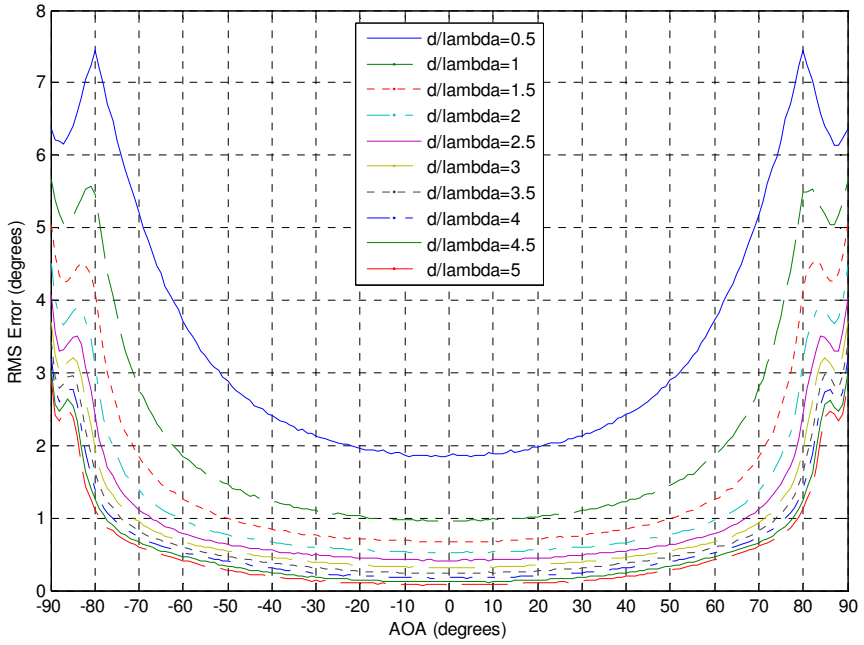

Figure 3.29 Phase Comparison RMS Error vs. AOA at d/λ value from 1 to 5 in steps of 0.5 (SNR = 20 dB)

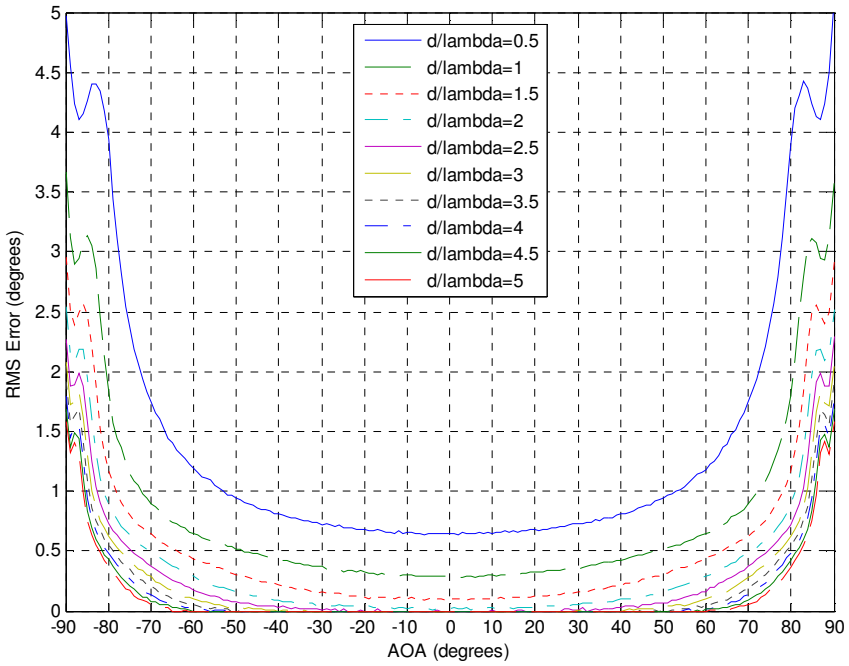

Figure 3.30 Phase Comparison RMS Error vs. AOA at d/λ value from 1 to 5 in steps of 0.5 (SNR = 30 dB)

Phase comparison algorithm has different performance characteristics over the AOA. Better DF accuracy is obtained for the signals arriving around boresight of the antenna. As the angle between the boresight of the antenna and the propagation vector of the signal increases, RMS error increases. For AOA greater than 60 degrees, the RMS error curve has very steep increasing characteristics. If more DF accuracy is required, it is possible to diminish azimuth coverage from 180 degrees to approximately 120 degrees. In this configuration, to cover a greater azimuthal range, more than one interferometer systems can be used to operate together.

To understand the overall DF accuracy of the experimental results, the statistical information, mean of the RMS error as a function of d/λ is plotted in Figure 3.31 for different SNR values.

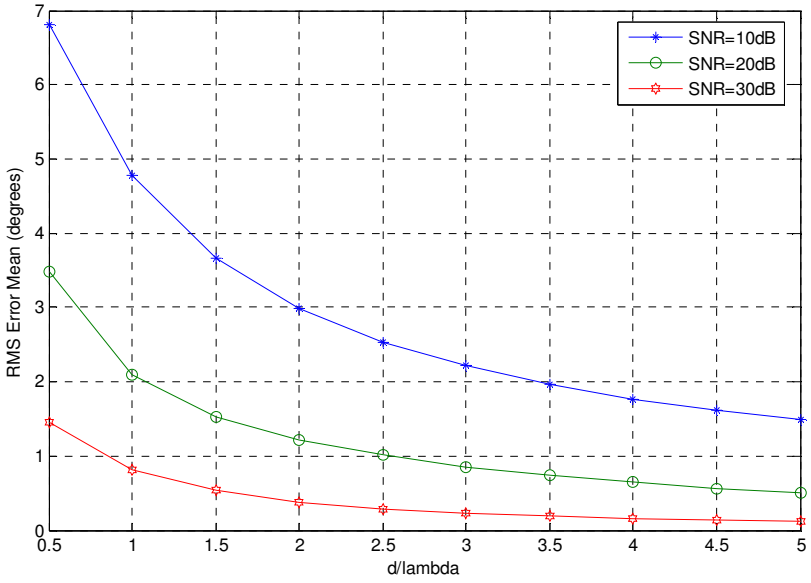

Figure 3.31 Mean of RMS Error vs. d/λ Values for SNR=10dB, 20dB and 30dB

As d/λ increases, mean and the standard deviation of the RMS error decreases and provide a better angular location. For high SNR values, as expected the algorithm has a better DF accuracy. For 30dB SNR, ignoring both ends of the entire azimuth

range where RMS error is large, an RMS accuracy of 1 degree RMS error mean is reached for the d/λ values greater than 1.

In phase comparison systems, one of the most challenging problems is to determine the baseline spacing according to the operating frequency band. The DF accuracy is directly proportional to the ratio of baseline spacing to the wavelength of the signal. Therefore the baseline spacing is determined upon the specifications and the needs of the system.

In this study, the operating frequency band is determined to be between 2 GHz and 6 GHz corresponding to a maximum wavelength of 15 cm and minimum wavelength of 5 cm. Since d/λ value must be greater than ½, the minimum baseline spacing of the antennas is limited to 7.5 cm.

For different baseline spacing corresponding d/λ intervals are given by

$1/2 < d/\lambda < 3/2$, for $d_1 = 7.5$ cm,

$1 < d/\lambda < 3$, for $d_2 = 15$ cm,

$3/2 < d/\lambda < 9/2$, for $d_3 = 22.5$ cm.

For the simulation, the baseline spacing is chosen to be to 15 cm and the frequency is varied between 2 GHz and 6 GHz. Here d/λ is varied between one and three proving a fair DF accuracy, mostly below RMS error of 2 degrees. Software simulations were performed to compare with the hardware implementation results. Software simulation results are plotted in Figure 3.32.

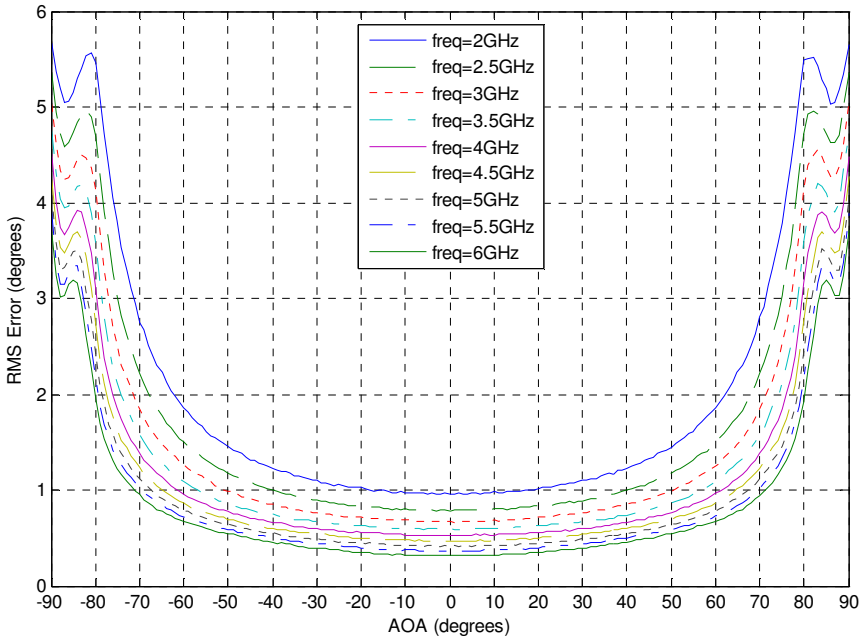

Figure 3.32 RMS Error vs. AOA when Frequency is varied from 2 GHz to 6 GHz in steps of 500MHz at SNR = 20dB.

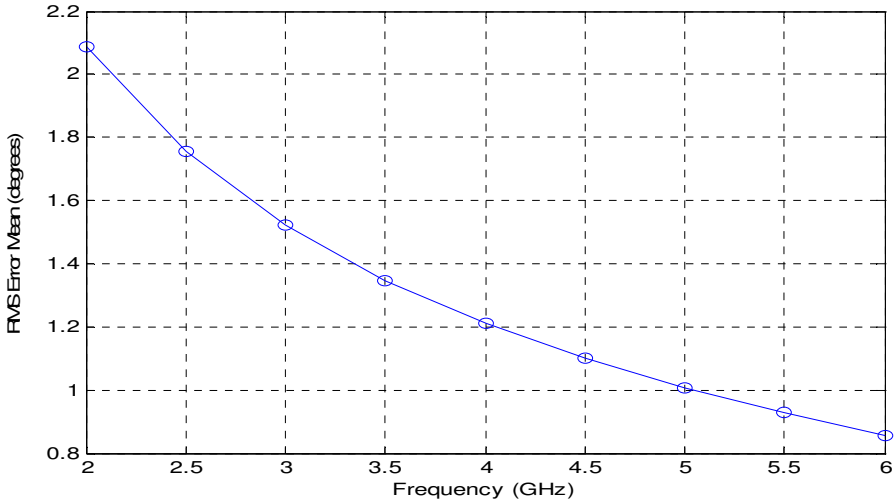

Figure 3.33 Mean of the RMS Error vs. Frequency at SNR=20dB.

In order to determine the search interval, prior information of AOA provided by another DF system must be available. In the simulations, the provided AOA is

assumed to be accurate, since main purpose is to calculate DF accuracy of the phase comparison system.

In order to decide the search interval, some cases are investigated. First, poor DF accuracy is obtained for small d/λ values and for the angles approaching ±90 degrees (Figure 3.32). To the ends of the entire angle range, the interval length is limited; therefore it should not be the main consideration. Considering the case when d/λ is equal to 1, the RMS error is calculated as 3 degrees around 70 degrees at 20dB SNR. The interval, in which the estimates reside 99.9 percent of probability, is obtained in reference [10] as (3.291 x 3) degrees. Hence it is good to consider search interval of 10 degrees in software simulations. Secondly, the angle difference of the maximum of lobes that makes ambiguity must be taken into consideration. When d/λ is equal to 3, maximum number of lobes are present in the calculated ML function. According to Eq.(3-6), phase difference is

$$\theta + 2\pi(k-1) = 2\pi.3.\sin\varphi_k, \quad \text{where k=1, 2, 3}$$

The φ_k's that are satisfying the above condition are the solutions. Solutions are the points which create ambiguity in the ML function. The minimized solution set of the condition is given by

$$\sin\varphi_2 - \sin\varphi_1 = 1/3, \text{ where } -90 < \varphi < 90$$

For this case, solutions, minimizing the search interval, are calculated ±9.5941 degrees. Therefore the angle between two lobes makes at least 19.1881 degrees, which is greater than defined search interval. Therefore for the simulations $\theta_{input} \pm 10$ degrees is a valid search interval.

Before proceeding to hardware implementation of the phase comparison algorithm, another point to investigate is the frequency resolution of the antenna gain patterns which will be stored in the RAM. Quantization errors due to frequency resolution must be evaluated to find an appropriate frequency resolution. Storing the antenna gain patterns dense over defined frequency interval requires much memory, whereas storing it sparse results in large quantization errors. Therefore, considering this trade-off, the quantization interval of the frequency should be determined carefully.

For frequency resolution of 100 MHz and 50 MHz, corresponding quantization errors over the entire frequency range are plotted in Figure 3.34 and Figure 3.35 respectively. In this case, the AOA of the source is set to 30 degrees and the simulation is performed when there is no noise and when SNR is equal to 20 dB.

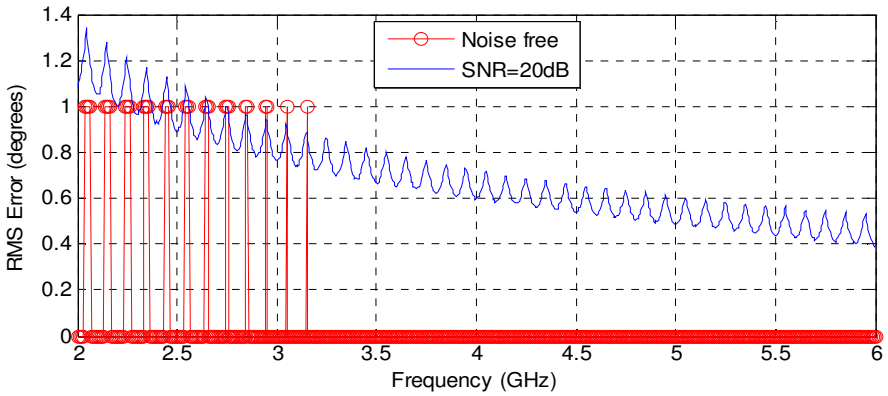

Figure 3.34 The RMS Error vs. Frequency when $f_{resolution}$ = 100 MHz (AOA=30 degrees)

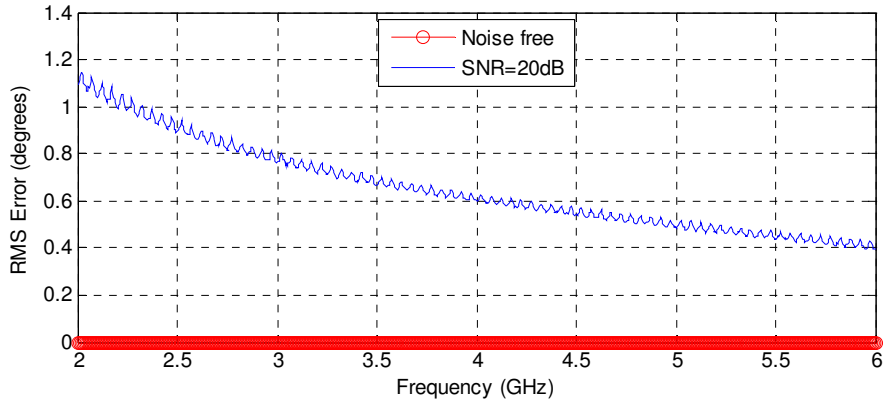

Figure 3.35 The RMS Error vs. Frequency when $f_{resolution}$ = 50 MHz (AOA=30 degrees)

As a result, the smaller the frequency resolution is used, the less quantization errors are obtained. For frequency resolution of 100 MHz, there exists a quantization error even at the noise free case. Therefore, frequency resolution of 50 MHz is more appropriate. However in the implementation it is better to set the resolution to a number which is a power of 2. In case of performing a division operation with fix point numbers, which takes more time and more resource, performing a shift

operation takes less resource and is faster. Therefore, the experiments are also performed for frequency resolution of 64MHz.

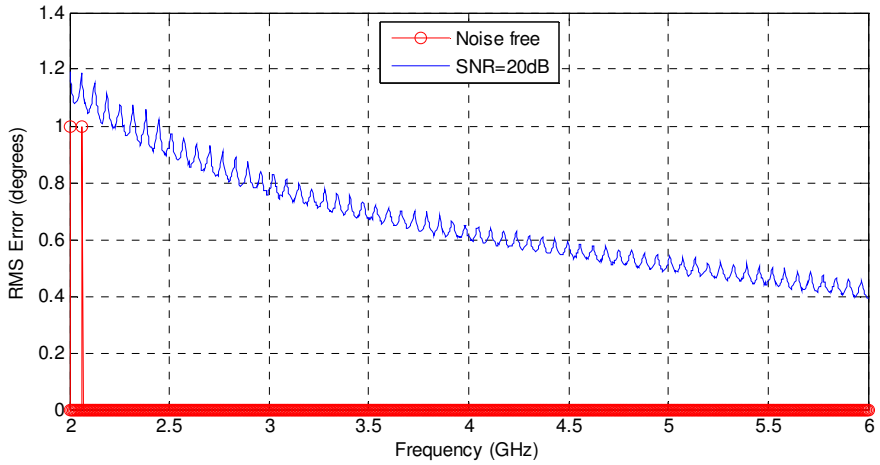

Figure 3.36 The RMS Error vs. Frequency when $f_{resolution}$=64 MHz (AOA=30 degrees)

According to Figure 3.36, the RMS error is smaller than 0.1 degrees for 20dB SNR and is getting smaller as the frequency is increasing over the entire frequency range.

3.2.2 Hardware Implementation

In this part, implementation of the phase comparison DF algorithm on an FPGA platform by using a commercial software tool, Xilinx System Generator Tool (Appendix B) will be presented. After implementation of the phase comparison algorithm, the part of the PDW Generator, which performs I and Q demodulation and delivers the complex amplitude to the phase comparison block, is implemented.

3.2.2.1 Hardware Design

In hardware implementation of the phase comparison algorithm, the inputs to the designed core are;

- the real and imaginary parts of the signals from each channel,
- the frequency of the received signals,
- the estimate of the AOA that another DF system provides.

The output of the hardware implementation is the AOA estimate of the emitter.

In the hardware design, the implementation of the phase comparison DF algorithm consists of 4 main blocks. These blocks are;

- PC_Angle_Counter, which specifies a valid search interval.
- PC_Address_Decoder, which determines the memory address interval
- PC_Function_Calculation, which computes the ML function.
- PC_AOA_Estimator, which estimates the AOA.

The purpose of the each implemented block is presented in the following parts of the chapter. The block diagram of the hardware implementation is given in Figure 3.37.

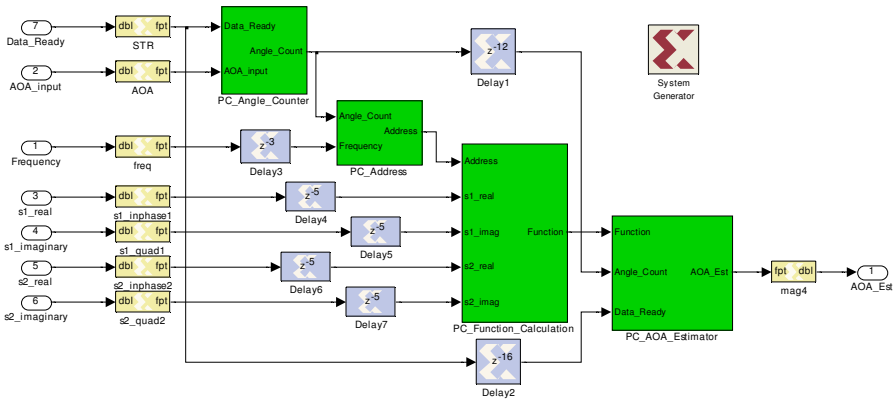

Figure 3.37 Block Diagram of Phase Comparison Implementation

3.1.2.1.4 PC_Angle_Counter Block

The purpose of this block is to determine the search interval of the algorithm. The inputs to this block are *Data_Ready* and *AOA_input* signals. *Data_Ready* signal gives information that a new data is ready for search interval calculation. *AOA_input* is the AOA value, which another DF system provides. The output is the search interval, *Angle_Count*, whose center is *AOA_input*.

In the design, *Data_Ready* signal is defined to be Boolean. It is assumed that *AOA_input* is in the interval between 90 and -90 degrees. It is represented as 8 bit signed integer. The output *Angle_Count* will be used for determining the memory address in the next block. Therefore it varies between 0 and 180 as an 8 bit unsigned integer.

The logic diagram of the *PC_Angle_Counter* block is given in Figure 3.38.

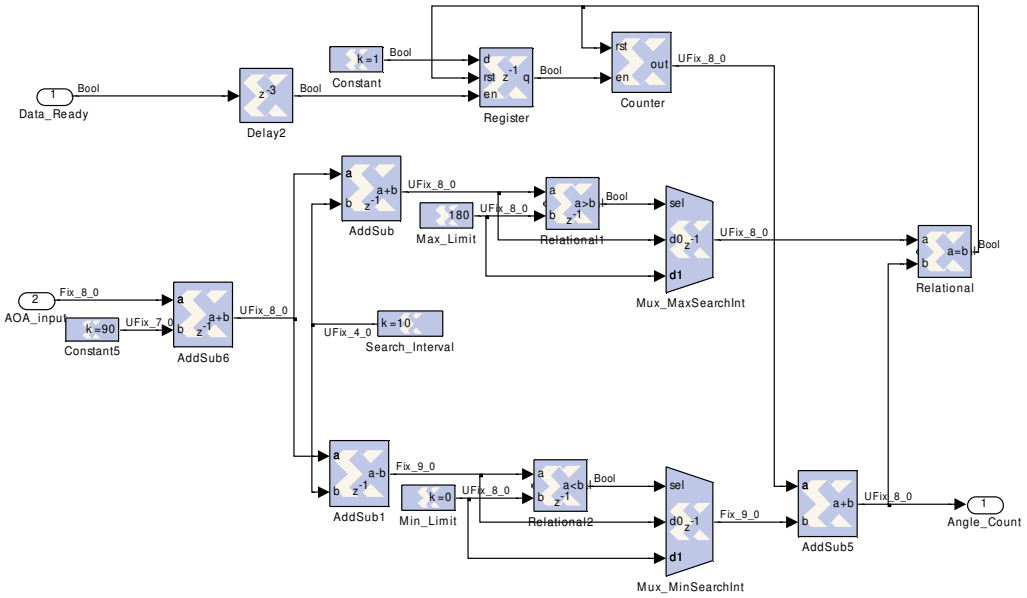

Figure 3.38 Logic Diagram of the PC_Angle_Counter Block

3.1.2.1.5 PC_Address_Decode Block

The purpose of this block is to calculate the memory address of antenna gain pattern. The inputs to this block are the *Frequency*, frequency of the received signal and the *Angle_Count*, corresponding angle count of the search interval. The logic diagram of the *PC_Adreess_Counter* block is given in Figure 3.39.

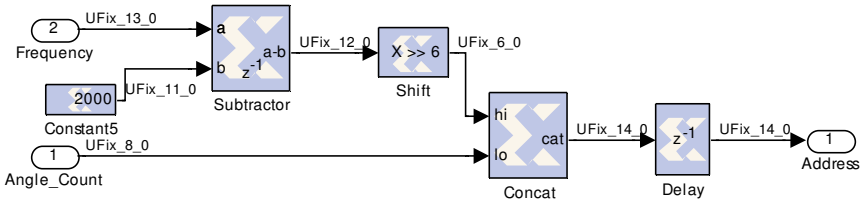

Figure 3.39 Logic Diagram of the PC_Address_Decoder Block

The frequency range is defined between 2GHz and 6GHz. The antenna gain patterns are stored in ROM block at frequencies changing in steps of 64MHz. It makes 63 steps to cover the 4GHz range, which can be represented with 6 bits. Therefore the total memory address is defined by 14 bits, resulting in memory depth of 16 K.

Memory address is formed up by concatenation of two signals. *Angle_Count,* input to the *lo* port of Concat Block, will occupy the least significant bits (LSB) of the memory address. The frequency signal, which is input to the *hi* port, will occupy the most significant bits (MSB) of the memory address.

In order to obtain minimum the quantization errors, antenna gain patterns are quantized at frequencies according to the given equation below.

$$f_{quantized} = 2032 + 64k \text{ where } (k=0, 1, 2, \ldots, 62)$$

3.1.2.1.6 PC_Function_Calculation Block

The purpose of this block is to calculate the ML function $J'(\phi)$ in Eq.(2-29). The inputs to the block are the real and imaginary parts of the received signals, s_1 and s_2, and the memory address. The logic diagram of the *PC_Function_Calculation* block is given in Figure 3.40.

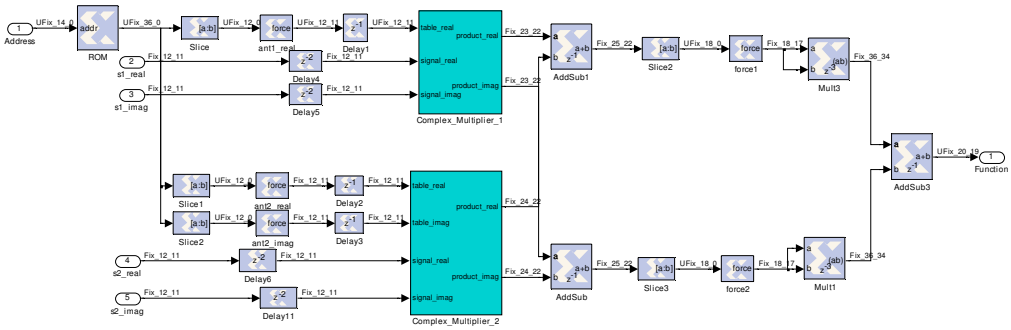

Figure 3.40 Logic Diagram of PC_Function Calculation Block

In order to find the AOA of the incoming signal, the algorithm tries to find the argument that maximizes the ratio given in Eq.(2-30), which is rewritten below for convenience.

$$\phi = \arg\max_{\phi}\left(\frac{\left|\mathrm{sg}^{*}(\phi)+\mathrm{dh}^{*}(\phi)\right|^{2}}{|\,g(\phi)|^{2}+|\,h(\phi)|^{2}}\right) \qquad \textbf{(2-30)}$$

For the implementation of the algorithm, some simplifications are made on the equation in order to minimize the number of operations. This will result in faster process time of the implemented algorithm. Taking the square root of the Eq.(2-30) does not affect the result of the algorithm. Therefore the algorithm becomes

$$\phi = \arg\max_{\phi}\left(\frac{\left|\mathrm{sg}^{*}(\phi)+\mathrm{dh}^{*}(\phi)\right|}{\sqrt{|\,g(\phi)|^{2}+|\,h(\phi)|^{2}}}\right) \qquad \textbf{(3-7)}$$

Using 64 MHz frequency steps, the antenna gain patterns of two antennas are stored in the memory. The antenna patterns are defined as

$$g_{stored} = \frac{g^{*}(\phi)}{\sqrt{|g(\phi)|^2 + |h(\phi)|^2}} \quad \textbf{(3-8)}$$

$$h_{stored} = \frac{h^{*}(\phi)}{\sqrt{|g(\phi)|^2 + |h(\phi)|^2}} \quad \textbf{(3-9)}$$

The Eq.(3-7) is simplified in order to find the maximum of $|sg_{stored} + dh_{stored}|$ function. As explained in Chapter II, h_{stored} is complex valued; therefore real and imaginary parts are stored separately. Real and imaginary parts are normalized between 1 and -1 interval. The other antenna pattern, g_{stored} is purely real; therefore only real part is stored in memory.

Since the signal and table values are complex valued, two blocks are designed for performing complex multiplication operation. Then the complex products of the designed blocks are added together separately as real and imaginary parts. Finally the magnitude of the products is calculated as an output of the block.

The logic diagrams of the *Complex_Multiplier_1* and *Complex_Multiplier_2* blocks are given in Figure 3.41 and Figure 3.42 respectively.

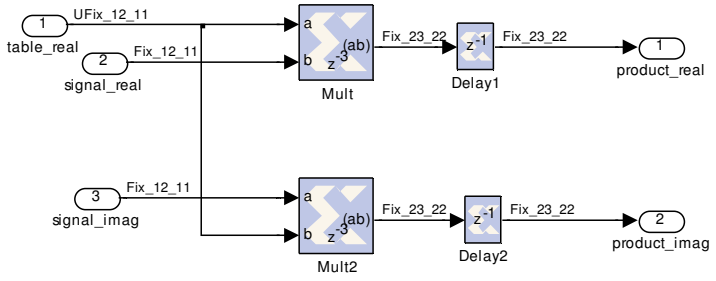

Figure 3.41 Logic Diagram of Complex_Multiplier_1 Block

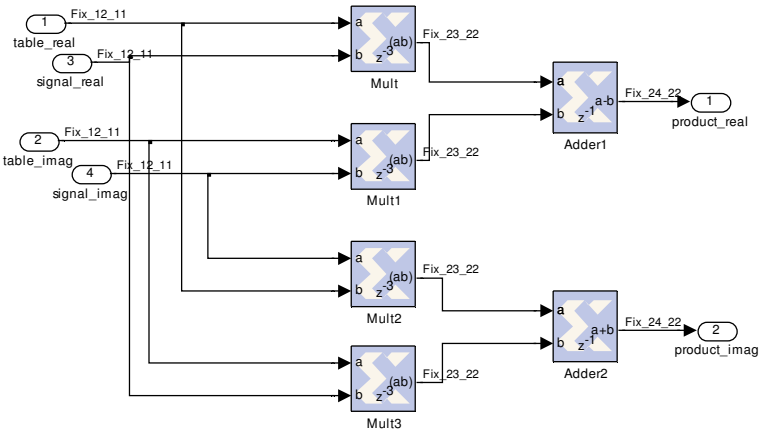

Figure 3.42 Logic Diagram of Complex_Multiplier_2 Block

3.1.2.1.7 PC_AOA_Estimator Block

The aim of this block is to find the angle at which maximum of the function occurs over the search interval. The inputs to the block are *Data_Ready*, *Angle_Count* and *Function* signals. At the end of the search interval, estimated AOA value, *AOA_Est*, is presented as an output of the block. The logic diagram of the PC_AOA_Estimator block is given in Figure 3.43.

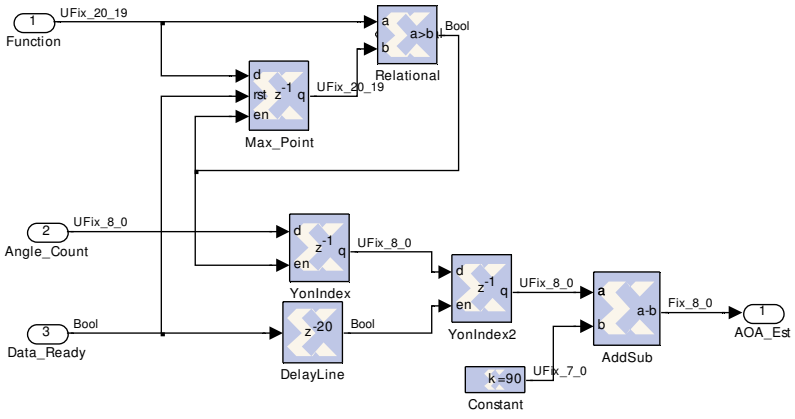

Figure 3.43 Logic Diagram of the PC_AOA_Estimator Block

To synchronize the *Correlation*, *Angle_In* and *Data_Ready* signals, these signals are delayed with proper delay blocks. After *Data_Ready* strobe is high, 39 clock cycles later *AOA_Est* is driven as an output. The latency is as follows; 3 clock cycles in *PC_Angle_Counter* block to determine the search interval, 2 clock cycle to find the appropriate address in *PC_Address* block, 12 clock cycles in *PC_Function_Calculation* block to calculate the ML function and 22 clock cycles in *PC_AOA_Estimator* block to find the angle at which maximum of the ML function occurs.

3.2.2.2 Simulation Results

Similar to the amplitude comparison hardware implementation, the number of bits for representing the complex antenna gain values is set to 12. Likewise, the real and imaginary parts of the pulse amplitude are set to 12 bits. Antenna displacement was chosen as 15 cm. The operating frequency range is selected between 2 GHz and 6 GHz.

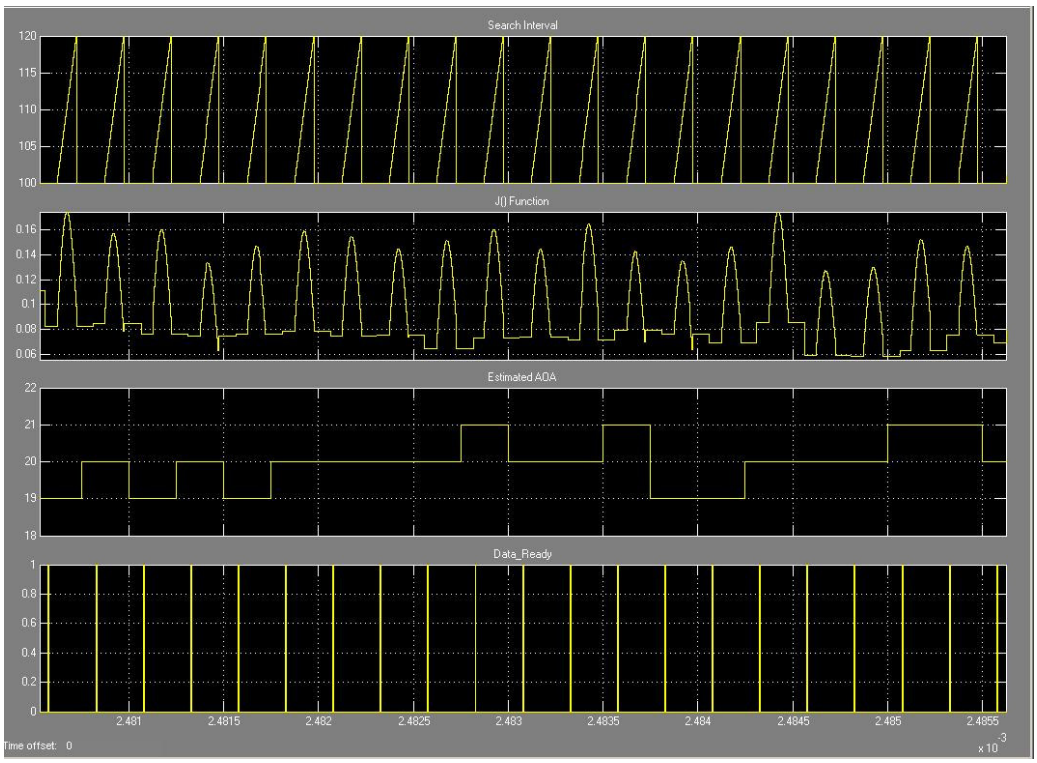

Figure 3.44 Screen Shot from System Generator (Implementation of Phase Comparison Algorithm)

In Figure 3.44 screen shot of the scope in hardware implementation is plotted. The pulses are generated when SNR value is set 20dB and pulse repetition interval (PRI) is set to 250 ns. For each pulse the search interval is defined around 20 degrees, which is AOA of pulse in this case. The AOA is estimated according to the calculated J( ) function.

The simulations of the implementation were performed at frequencies from 2 GHz to 6 GHz in steps of 1 GHz at 20dB SNR. $5x10^3$ simulations are performed for each AOA from -90 to 90 degrees. RMS error vs. AOA curves corresponding to different frequencies are plotted in Figure 3.45.

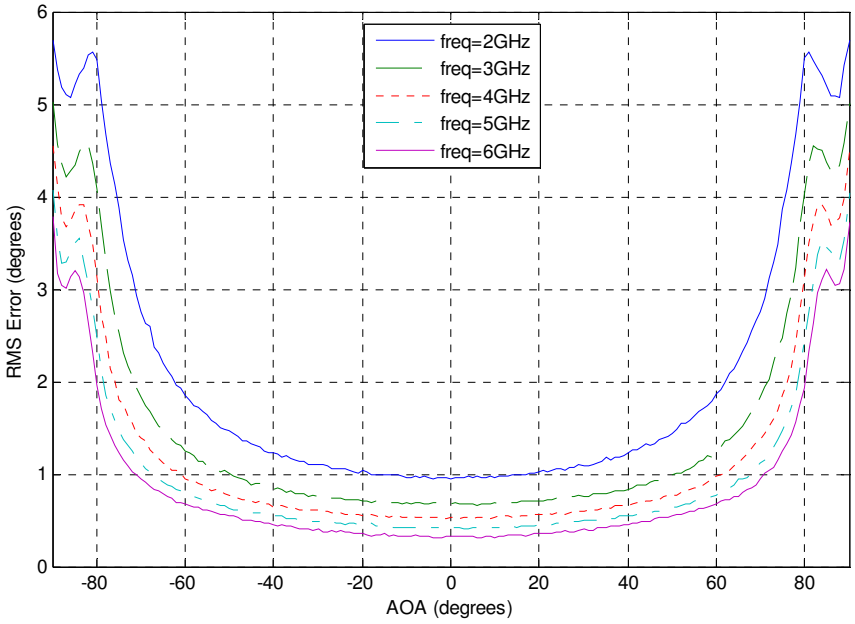

Figure 3.45 Phase Comparison Hardware Implementation RMS Error vs. AOA (SNR=20dB)

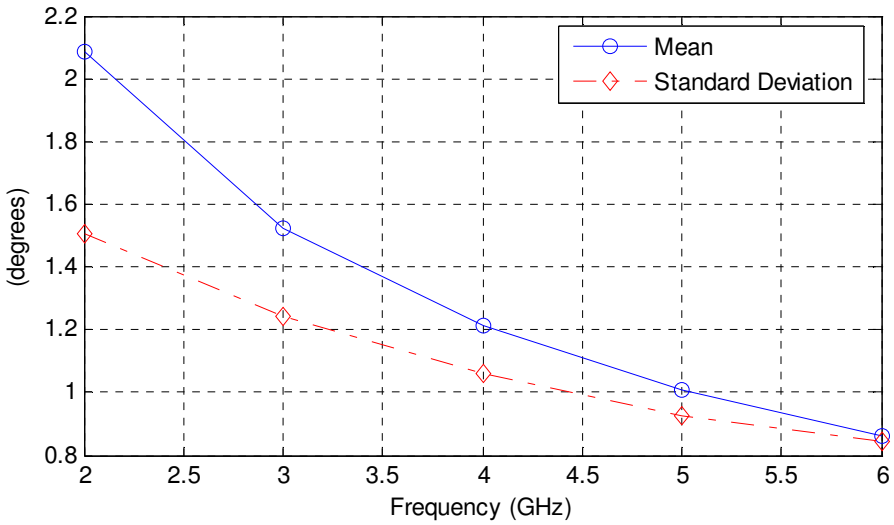

Figure 3.46 Mean and The Standard Deviation of RMS Error

To investigate the quantization errors, hardware simulations are performed. Compared to higher frequencies, at lower frequencies the quantization errors are greater due to linear scaling of the antenna gain table. Therefore in order to see the quantization effects, simulations are performed when the operating frequency is varied between 2000 MHz and 2512 MHz in steps of 8 MHz and AOA is 30 degrees at 20 dB SNR. The mean of the estimated AOA is given in Figure 3.47.

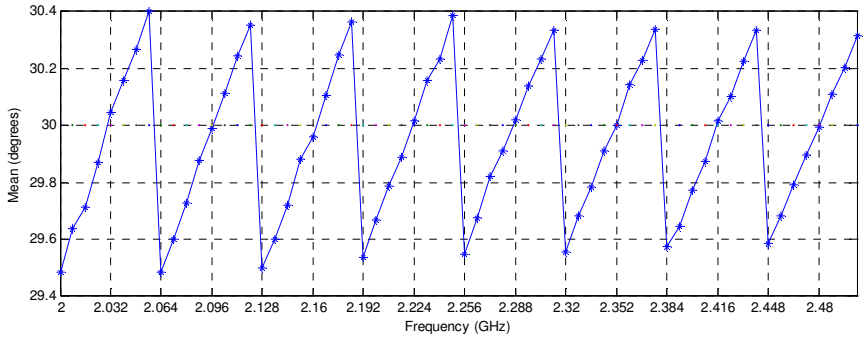

Figure 3.47 Mean of the Estimated AOA vs. Frequency

The RMS Error of the hardware implementation due to quantization of frequency is calculated and is compared with software simulation results in Figure 3.48.

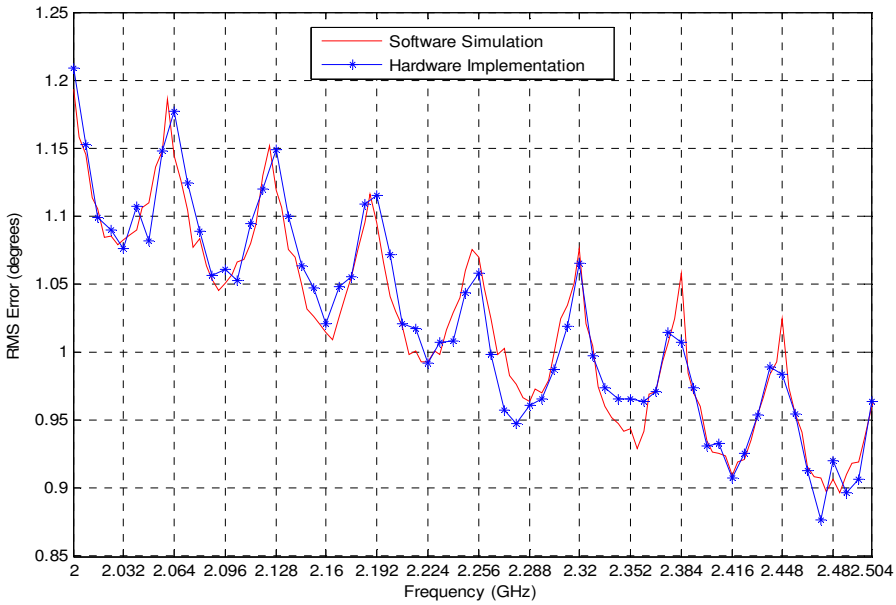

Figure 3.48 RMS Error of Hardware Implementation due to Frequency Quantization

The hardware resources, which are utilized for the phase comparison algorithm implementation, are given in Table 3-5.

Table 3-5 FPGA Resource Utilization Summary of Phase Comparison Implementation

Number of Slices	555
Number of Flip Flops	863
Number of 4 input LUTs	410
Number of External IOBs	78
Number of MULT18X18s	8
Total equivalent gate count for design	2,147,733

According to timing analysis, the following results are obtained.

Requested Constraint : 5.000 ns

Actual Constraint : 4.719ns

The number of signals not completely routed for this design is: 0 ns.

The average connection delay for this design is: 0.908 ns.

The maximum pin delay is: 4.079 ns.

The average connection delay on the 10 worst nets is: 3.781 ns.

Listing Pin Delays by value: (ns)

d < 1.00	< d < 2.00	< d < 3.00	< d < 4.00	< d < 5.00	d >= 5.00
2813	1056	169	69	4	0

According to synthesis report, all constraints were met and all signals are completely routed.

3.2.2.3 Part of PDW Generator Design

After evaluating test results of the hardware implementation of the phase comparison algorithm, the part of the PDW generator, which extracts the pulse amplitude of the received signals, is implemented and fitted before the phase comparison algorithm. In mathematical formulation and implementation, complex valued signals are used. However in practical usage it is not likely to extract complex representations of each pulse or continuous wave at the time of sampling. Moreover it is not very usable, since it is vulnerable to affects of noise or incorrect measurements very much. Therefore it is more acceptable to represent the received signal per pulse basis or for CW signals for a period of time.

The hardware implementation consists of three main blocks. These are;

- I_Q_Demodulator, which performs I and Q demodulation.
- Magnitude_Calculator, which calculates the magnitude of the signal.
- Phase_Comparison, which performs phase comparison algorithm.

In the simulations the signals are sampled at a rate of 200 MHz. The block diagram of the hardware implementation is given in Figure 3.49.

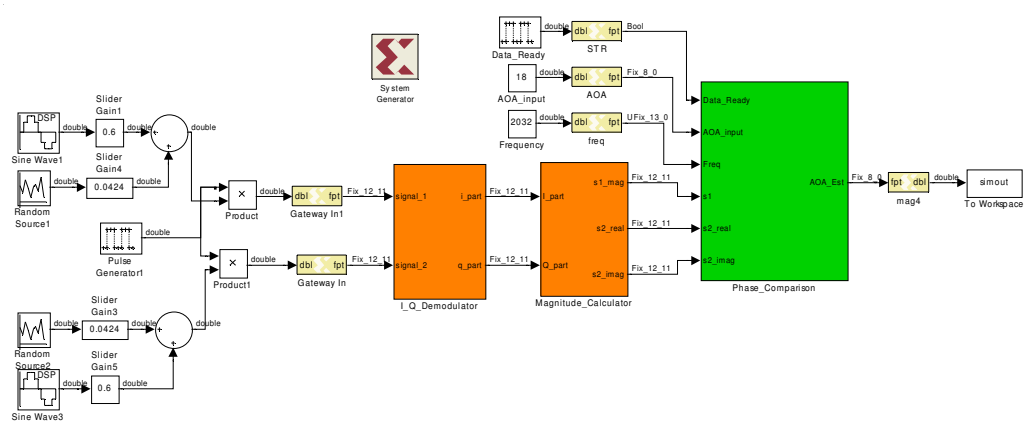

Figure 3.49 Block Diagram of Phase Comparison Implementation

3.2.2.3.5 I_ Q_Demodulator Block

The signals received by the antennas are real valued, and it is assumed that there is no modulation in the pulse or in the continuous wave. Then the received signals are defined by

$$s_1 = A\sin(wt) \tag{3-10}$$

$$s_2 = A\sin(wt+\varphi) \tag{3-11}$$

where A denotes the pulse amplitude, w is the operating frequency, t represents time and φ is the phase difference. Demodulation of signals is often done by obtaining in-phase and quadrature components. There are several methods to decompose the signal into these components [8]. In this case one of the received signals is used as a local oscillator. The block diagram of I and Q demodulation process is given Figure 3.50.

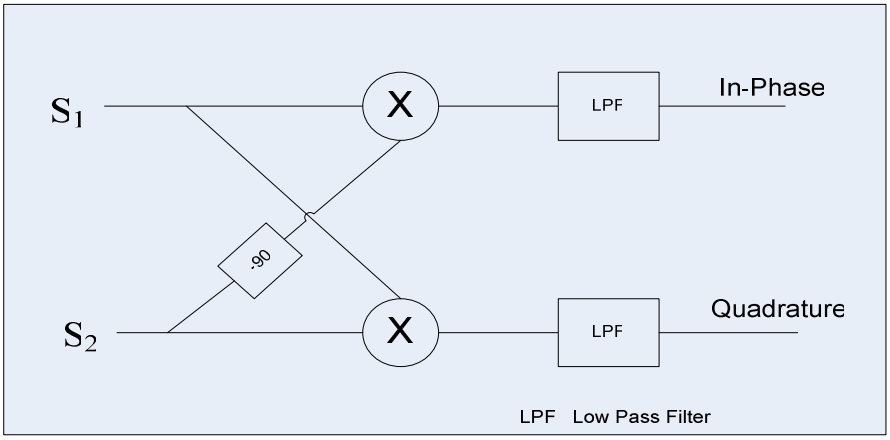

Figure 3.50 I and Q Demodulation

After the I and Q demodulation is applied, the in-phase and quadrature components are obtained as

$$I = A.\sin(wt)A.\sin(wt+\varphi) = A\frac{1}{2}[\cos(2wt+\varphi)+\cos\varphi] \xrightarrow{LPFilter} \frac{A}{2}\cos\varphi \qquad \textbf{(3-12)}$$

$$Q = A.\sin(wt)A.\sin(wt+\varphi-90) = A\frac{1}{2}[\cos(2wt+\varphi-90)+\cos(\varphi-90)] \xrightarrow{LPFilter} \frac{A}{2}\sin\varphi \qquad \textbf{(3-13)}$$

According to Eq.(3-12) and Eq.(3-13), after I and Q demodulation process, the amplitude and phase difference information of original incoming signal is preserved, whereas frequency is information is lost. The logic diagram of the implementation of I and Q demodulation is given in Figure 3.51.

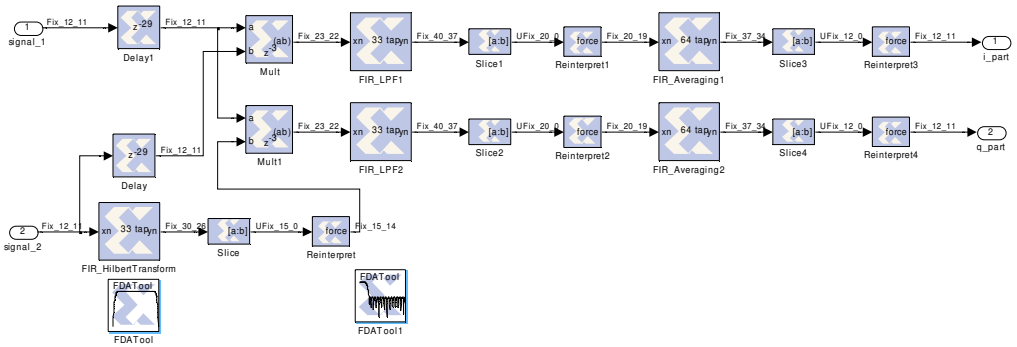

Figure 3.51 Logic Diagram of I and Q Demodulator Block

In order to lag the second signal by 90 degrees, an FIR filter is designed. The filter performs a Hilbert Transformation, which is an all pass and performs 90 degree phase shift. The filter is designed to pass the signals between 10 MHz and 90 MHz. The magnitude response of the 33 tap filter is shown in Figure 3.52.

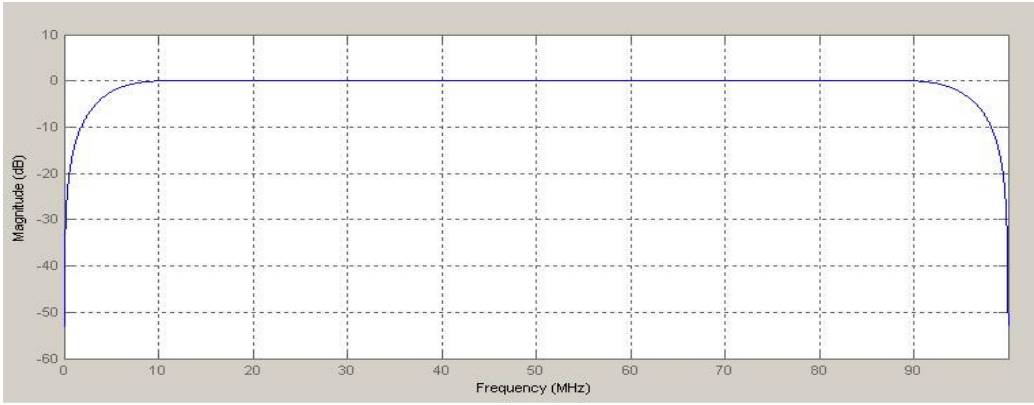

Figure 3.52 Magnitude Response of FIR Filter (Hilbert Transformation)

According to Eq. (3-12) and Eq. (3-13), the signals are exposed to a low pass filter (LPF) to eliminate higher order frequencies. The filter is designed to pass the signal that has a frequency smaller than 10 MHz. The magnitude of the 33 tap LPF is shown in Figure 3.53.

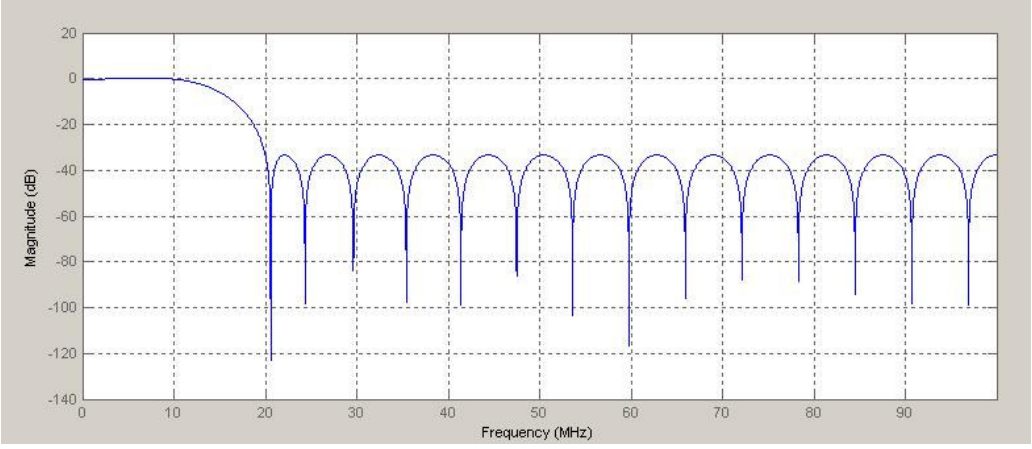

Figure 3.53 Magnitude Response of FIR LPF

After the signals are filtered through LPF, they are averaged by another FIR filter. In the implementation the FIR filter has 64 taps with constant coefficients. Alternatively more or less filter taps can be utilized depending on the pulse width of the received signal.

3.2.2.3.6 Magnitude_Calculator Block

The inputs to this block are in-phase and quadrature components and the output is the amplitude of the received signals. For the output, the real part of the signal is represented by in-phase component; on the other hand the imaginary part is represented by quadrature component. The other input is accepted as a reference and only magnitude of the signal is calculated since the phase difference is contained in the first output. The rearranged signals are given by

$$s_1' = \frac{A}{\sqrt{2}} \tag{3-14}$$

$$s_2' = \frac{A}{2}\cos\varphi + j\frac{A}{2}\sin\varphi \tag{3-15}$$

The logic diagram of the *Magnitude_Calculator* block is given in Figure 3.54.

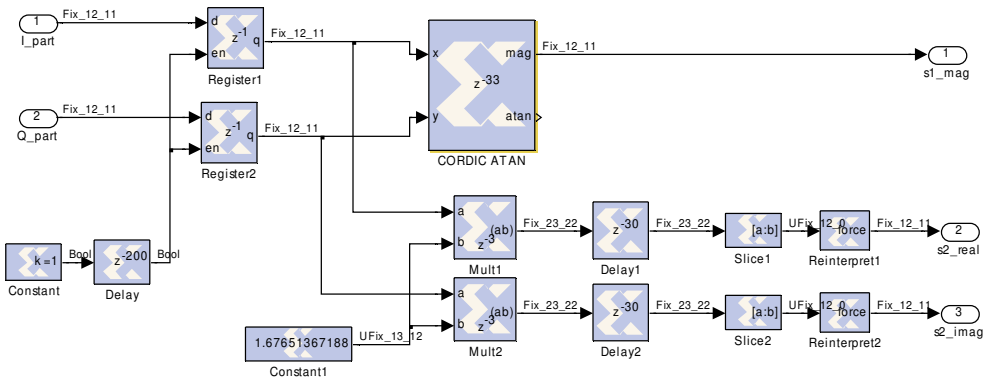

Figure 3.54 Logic Diagram of Magnitude_Calculator Block

3.2.2.4 Simulation Results

The output of the PDW generator block is driven to the phase comparison block to locate the angular position. Since one of the input signals is represented as a pure real signal, a minor modification is made in the hardware implementation. The imaginary part of s1, which is 0, is removed. The modified block diagram of the phase comparison implementation is given in Figure 3.55.

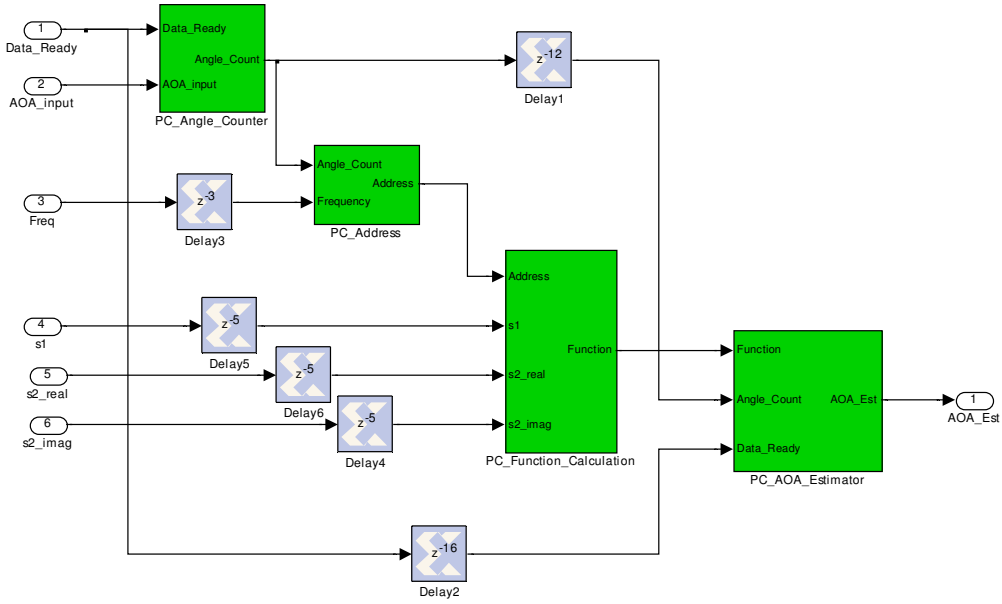

Figure 3.55 Block Diagram of the Phase Comparison Implementation

Several simulations are performed under various conditions to have information about the DF accuracies of the phase comparison algorithm. In order to see the affects of i and q demodulation and quantization, simulations are performed once again. Since adequate simulation results are obtained for the hardware implementation of phase comparison algorithm, in this part limited simulations will be performed and compared to previous results.

In the simulations, sampling rate is set to 200 MHz and AOA chosen as 20 degrees. White Gaussian noise, whose variance is equal to 1 and mean is 0, is generated and is added to signal to obtain a SNR value of 20 dB. Simulation results are given in Table 3-6.

Table 3-6 Mean and Standard Deviation of RMS Error of Estimated AOA

Frequency	Mean (degrees)	Standard Deviation (degrees)
2032 MHz	20.05	1.176
2992 MHz	20.003	0.876
4016 MHz	20.002	0.530
5040 MHz	19.997	0.451
6000 MHz	20.001	0.458

According to the Table 3-6, the mean of the estimated AOA values are very precise. On the other hand, the standard deviation of the estimations is slightly larger than the expected results, which are obtained in Chapter 3.2.3. There exist some reasons beneath this fact. First, the signals are exposed to filters in I and Q demodulation phase, where some precision is lost in the process. Secondly, between these steps the signals are truncated and the number of bits is decreased since they grow up large. Lastly, while finding the magnitude of the in-phase and quadrature components, again some precision is lost. In short mostly the error occurs due to quantization. On the other side in I and Q demodulation process, the outputs are averaged with a 64 tap FIR filter. Clearly, averaging the outputs minimizes the affect of noise. Regarding these points, the DF accuracy is calculated to be slightly larger than the hardware implementation results.

CHAPTER IV

CONCLUSIONS

In this study, two well-known monopulse direction finding techniques, namely amplitude comparison and phase comparison algorithms are implemented on an FPGA platform. In both algorithms mathematical modeling is done using maximum likelihood approach.

The amplitude comparison system consists of a circular array of six equally spaced antennas. Two approaches are proposed to locate the angular position of the emitter using the amplitude comparison algorithm. In the first approach pulse amplitudes received from all of the antennas are used. Whereas in the second approach, first the antenna that receives the maximum pulse amplitude is determined; then the pulse amplitudes received from the antennas, which are adjacent to this antenna, are determined. These three pulse amplitudes are taken into consideration and used in the DF algorithm. In both approaches the boresight of the antenna, which receives the maximum pulse amplitude, is selected to be the center of the search interval of 120 degrees. Over this search interval, AOA is estimated depending on the ML function. According to simulation results, utilizing only three pulse amplitudes instead of pulse amplitudes from all channels provides a better DF accuracy at low SNR values. As the SNR value increases both approaches have similar DF accuracies.

The phase comparison system considered consists of a linear phase array of two omni directional antennas. One of the most important parameter in a phase comparison system is the ratio of the antenna baseline spacing to the wavelength of the received signal, since it is directly related to the number of ambiguous points. In order to resolve ambiguity problem, the prior information of AOA is input to the algorithm. This AOA is taken as a center of a search interval. Over this search

interval, the ML function depending on complex pulse amplitudes and complex antenna gains are calculated. Consequently AOA is estimated by examining the ML function.

The primary aim of the thesis is to evaluate both of the algorithms and to implement them on an FPGA platform. Target platform is selected to be FPGA, since FPGA is widely used in DSP and digital communication areas. The advantage of using FPGA is that highly parallel structures can be employed when compared to DSP chips or microprocessors. By using parallel structures, throughput can be easily increased according to design specifications. Therefore the throughput is not limited directly by the core clock, but limited by the FPGA resources. Hence while implementing the algorithms, the performance and the resource allocation trade-off must be taken into consideration.

In implementation phase of the algorithms, some limitations and modifications were considered to meet the desired results. First, the algorithms were modified and optimized in order to have a better performance with less complexity and less hardware resources while not distorting the DF accuracy of the algorithms. The form of antenna gain tables stored in RAM was modified to avoid the operations like taking square root, multiplying with a constant or division in FPGA. The operations like division and taking square-root require more hardware resources and have long cycles of process time. Therefore the results may contain large quantization errors. Hence, in the hardware design these operations were avoided as much as possible. Another important point is the number of bits required for each signal to be represented in the FPGA. The number of bits used in the design is proportional to the resources used in FPGA. Number of bits in the algorithm was chosen to be 12 for representing the signal amplitudes and antenna gain tables. The calculated quantization errors were very small, since insignificant precision is lost regarding the FPGA resource allocation.

Hardware simulations of both algorithms were performed at a FPGA clock frequency of 200 MHz, which is equivalent to 5 ns of clock period. The amplitude comparison

algorithm implementation on FPGA is able to locate the angular position for 120 clock cycles, which means processing of nearly 1.6 MPPS. If more speed is needed according to system needs, the throughput can be increased, since FPGA platform designs are highly configurable and flexible. The amplitude comparison blocks can be used in parallel or some adjustments may be performed in algorithm like minimizing the search interval. For the phase comparison algorithm, it takes 20 clock cycles to determine the AOA of emitter, which means processing of nearly 10 MPPS. The same improvements, like using parallel structures or shortening the search interval if possible, are keys to minimize the process time to determine the AOA. Actually the process time of both algorithms is directly proportional to the defined search interval. The cost of using parallel structures is to allocate more resource in the FPGA.

The results of both amplitude comparison and phase comparison algorithm simulations indicate that the RMS error varies depending on the angular location of the emitter. Therefore mean of the RMS error is taken as an evaluation criterion at fixed SNR.

For the amplitude comparison algorithm, mean of the RMS error obtained in software simulations is nearly 5.2 degrees at 20dB SNR. Phase comparison algorithm has better DF accuracy when compared to amplitude comparison algorithm. When SNR is equal to 20dB and the baseline spacing is set to 15 cm, the mean of the RMS error alters between 0.9 and 2.1 degrees over the 2GHz and 6 GHz frequency range. The results of the hardware implementation of both algorithms in FPGA with fixed point arithmetic clearly demonstrated that the errors due to quantization are very small.

REFERENCES

[1] D. Curtis Schleher, "Electronic Warfare in the Information Age", Artech House, 1999.

[2] Evrim Anıl Evirgen, "Direction Finding With a Uniform Circular Array Placed on a Moving Platform", Ms. Thesis, Middle East Technical University, 2001.

[3] Arzu Tuncay Koç, "Direction finding with a Uniform Circular Array via Single Snapshot Processing", Ph.D. Thesis, Middle East Technical University, 1996.

[4] Samuel M. Sherman, "Monopulse Principles and Techniques", Artech House, 1984**.**

[5] Avionics Department of the Naval Air Warfare Center Weapons Division *"Electronic Warfare and EW Systems",* 1993.

[6] Stephen E. Lipsky, "*Microwave Passive Direction Finding"*, Wiley, 1987

[7] Ferhat Arıkan, "Design of a Monopulse Comparison DF processor Using A Maximum Likelihood Algorithm", Ms. Thesis, Middle East Technical University, 1992.

[8] Richard G. Wiley, *"Electronic Intelligence: The Analysis of Radar Signals"*, Artect House, 1993.

[9] Charles W. Therrien, *"Discrete Random Signals and Statistical Signal Processing",* Prentice Hall, 1992.

[10] Athanasios Papoulis, *"Probability, Random Variables and Stochastic*

Processes", McGraw-Hill Inc. 1991.

[11] *"Virtex-II Pro Platform FPGA Handbook",*Xilinx Inc, 2002.

[12] "*Xilinx System Generator for DSP v6.3 User Guide, A Tutorial Indroduction"*, Xilinx Inc, 2004.

[13] Steve Zack, Suhel Dhanani *,"DSP Co-Processing in FPGAs: Embedding High-Performance, Low-Cost DSP Functions",* Xilinx Inc, 2004.

[14] J. Hwang, B. Milne, *"System Level Tools for DSP in FPGAs"*, 2001.

[15] AISN Software, *"TableCurve 2D", 1994.*

APPENDICES

APPENDIX A

A.1 Antenna Gain Pattern

A software tool is utilized to generate the antenna gain pattern for the amplitude comparison method. The software tool [15] is capable of displaying the X-Y data along with the fitted equation, and optionally with confidence and/or prediction intervals, data references and a fit reference. The graph can be scaled and formatted. A great variety of output options like full numeric evaluation of the selected equations are available. The equation list is sorted by tool's goodness-of-fit sort criteria and its contents can be filtered to only contain certain types of equations.

The main design criteria considered for the generation of the antenna pattern are 3dB beamwidth, 10dB beamwidth and minimum gain of the antenna. The half-wave beamwidth of the antenna is determined to be 60 degrees. 10dB beamwidth is selected to be 120 degrees; and the minimum gain along the antenna pattern is set to 40dB. According to the design specifications, the antenna pattern is obtained by fitting a Gaussian curve, which is given by

$$y = a + \frac{b}{1 + e\left(\frac{(x - c)}{d}\right)^2} e^{-\frac{(1-e)}{2}\left(\frac{(x-c)}{d}\right)^2} \qquad \textbf{(A-1)}$$

where, a is equal to 9.9×10^{-05}, b is equal to 0.9999, c is equal to 180, d is equal to 28.7382 and e is equal to 0.4218.

The generated antenna pattern, which is plotted in logarithmic scale, is given in Figure A.1.

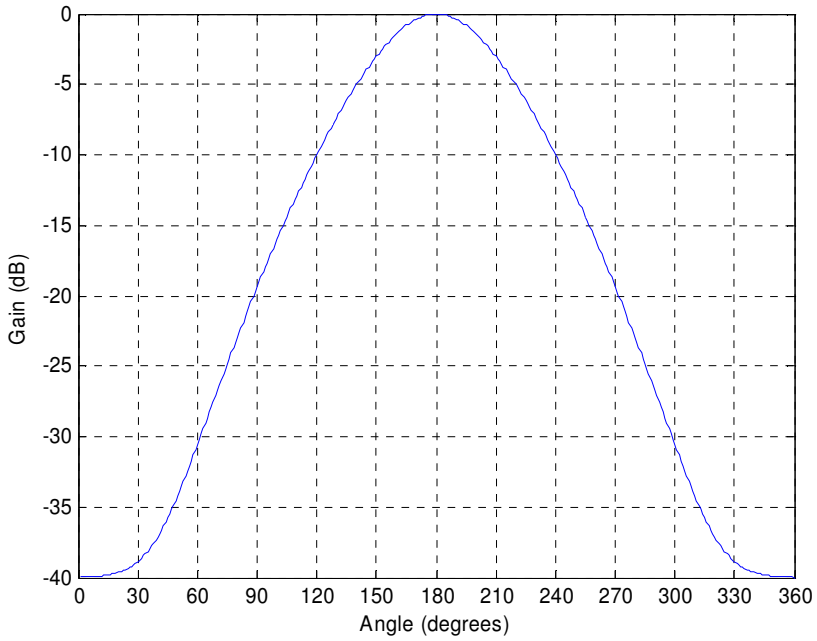

Figure A.1 Generated Antenna Pattern for Amplitude Comparison

APPENDIX B

B.1 FPGA's

A field programmable gate array (FPGA) is a general-purpose integrated circuit. It is "programmed" by the designer rather than the device manufacturer. Unlike an application-specific integrated circuit (ASIC), which can perform a similar function in an electronic system, an FPGA can be reprogrammed, even after it has been deployed into a system.

An FPGA provides the user with a two-dimensional array of configurable resources that are used to implement a wide range of arithmetic and logic functions. These resources include dedicated 18x18 bit multipliers, dual port memories, lookup tables (LUTs), registers, tristate buffers, multiplexers, and digital clock managers.

FPGAs are high performance data processing devices. FPGA performance is derived from the ability they provide to construct highly parallel architectures for processing data. Compared to a microprocessor or DSP processor, where performance is tied to the clock rate at which the processor can run, FPGA performance is tied to the amount of parallelism that can be brought to bear in the algorithms making up a signal processing system. Currently system frequencies of 100-200 MHz are commonly used today. A combination of increasingly high system clock rates and highly distributed memory architecture gives the system designer an ability to exploit parallelism in DSP applications that operate on data streams. For example, the raw memory bandwidth of a large FPGA running at a clock rate of 150 MHz can be hundreds of terabytes per second. [12]

There are wide ranges of DSP applications as digital up/down converters, digital FIR filters etc that can be implemented only in custom integrated circuits (ICs) or

specifically in an FPGA. Advantages of using an FPGA include significantly lower non-recurring engineering costs than those associated with a custom IC (FPGAs are commercial off-the-shelf devices), shorter time to market, and the configurability of an FPGA, which allows a design to be modified, even after deployment in an end application.

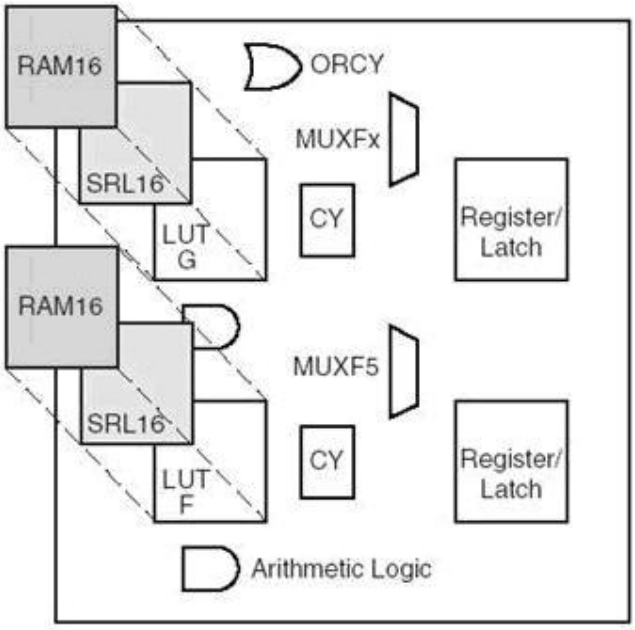

Figure B.1 Virtex Family FPGA Logic Slice

Logic Slice is the fundamental unit of the logic fabric array. It is instructive to take a closer look at the Virtex family logic slice construction, which is given in Figure B.1. Each logic slice contains two 4-input lookup tables (LUTs), two configurable D-flip flops, multiplexers, dedicated carry logic, and gates used for creating slice based multipliers. Each LUT can implement an arbitrary 4-input Boolean function. Coupled with dedicated logic for implementing fast carry circuits, the LUTs can also be used to build fast adder/subtractors and multipliers of essentially any word size. In addition to implementing Boolean functions, each LUT can also be configured as a 16x1 bit RAM or as a shift register (SRL16). An SRL16 shift register is a

synchronously clocked 16x1 bit delay line with a dynamically addressable tap point. In recent years, field-programmable gate arrays (FPGAs) have become key components in implementing high performance digital signal processing (DSP) systems, especially in the areas of digital communications, networking, video, and imaging. The logic fabric of today's FPGAs consists not only of look-up tables, registers, multiplexers, distributed and block memory, but also dedicated circuitry for fast adders, multipliers, and I/O processing (e.g., giga-bit I/O). The memory bandwidth of a modern FPGA far exceeds that of a microprocessor or DSP processor running at clock rates two to ten times that of the FPGA. Coupled with a capability for implementing highly parallel arithmetic architectures, this makes the FPGA ideally suited for creating high-performance custom data path processors for tasks such as digital filtering, fast Fourier transforms, and error correcting codes [14].

APPENDIX C

C.1 Development Tool Flow

Using MATLAB® and Simulink® from MathWorks, coupled with Xilinx System Generator™ for DSP, signal processing algorithms can be modeled, simulated, and verified specific to target hardware platform in the Simulink environment. When the simulation is run, various signals can be stored as variables in the workspace providing to make post simulation analysis.

The design flow typically involves the following steps: First hardware model is developed and verified by the using industry tools. Then, Xilinx System Generator generates an HDL circuit representation that is bit- and cycle-true meaning that the behavior is guaranteed to match the functionality seen in the simulation model. Finally, the Integrated Software environment (ISE) design tools synthesize the design and produce a bitstream that is used to program the FPGA. The error-prone and time-consuming phase, which is translating the hardware design into HDL, is automatically done by the software tool [13].

C.2 Xilinx System Generator

System Generator is a new system level tool built on top of Simulink. It facilitates DSP design for FPGAs. Simulink provides a convenient high level modeling environment for DSP systems, and consequently is widely used for algorithm development and verification. System Generator maintains an abstraction level while keeping the traditional Simulink blocksets. System Generator automatically translates designs into hardware implementations. The implementation is faithful in that the system model and hardware implementation are bit-identical and cycle-identical at sample times as designed in Simulink. The implementation is made efficient through the use of Intellectual Property (IP) cores that provide a range of

functionality from arithmetic operations to complex DSP algorithms. These cores have been carefully designed to run at high speed and be area efficient [14].

System Generator provides two main Matlab Simulink simulation libraries, the "Xilinx Blockset" and "Xilinx Reference Blockset". Each library contains blocks that are used to construct models in Simulink environment. These blocks provide arithmetic, logic, memory, and signal processing abstractions suitable for implementing DSP systems in an FPGA platform. In addition to these simulation libraries, System Generator provides implementations for each function, and code generation software for translating subsystems comprised of Xilinx blocks into hardware descriptions of the system model. The simulation models are bit-accurate to the hardware description, and at sample rates observable in Simulink, are also clock cycle accurate to the hardware. Consequently, the detailed hardware behavior can be understood from the Simulink simulation [12].

A System Generator model is constructed of blocks from the Xilinx blocksets, each of which contributes to the hardware description created by the code generator. One of the primary advantages of designing hardware using System Generator is the rich simulation framework Simulink provides to stimulate, debug, and analyze the executable model [12].

Each realizable signal which is to be translatable into hardware, must be sampled and be either a fixed-point or Boolean type. Since System Generator's goal is to model realizable digital systems, it does not support modeling of continuous time signals. Data types and sample rates are propagated through a System Generator model under normal Simulink propagation rules. This means that any source to a System Generator subsystem must be sampled and quantized. Signals passing from Simulink blocks to Xilinx blocks must go through Xilinx gateways, which enforce these constraints [12].

An output data type of Xilinx blocks can be *boolean*, *unsigned*, or *signed (two's complement)*, with a user-defined numerical precision. Figure C.1 shows a Xilinx Multiplier that forces outputs to be a 16-bit unsigned value, with 15 fractional bits.

Quantization parameter can be *truncate* or *round* (unbiased: /- Inf). In the user defined parameters overflow parameter must be selected one of *wrap*, *saturate* or *flag as error.*

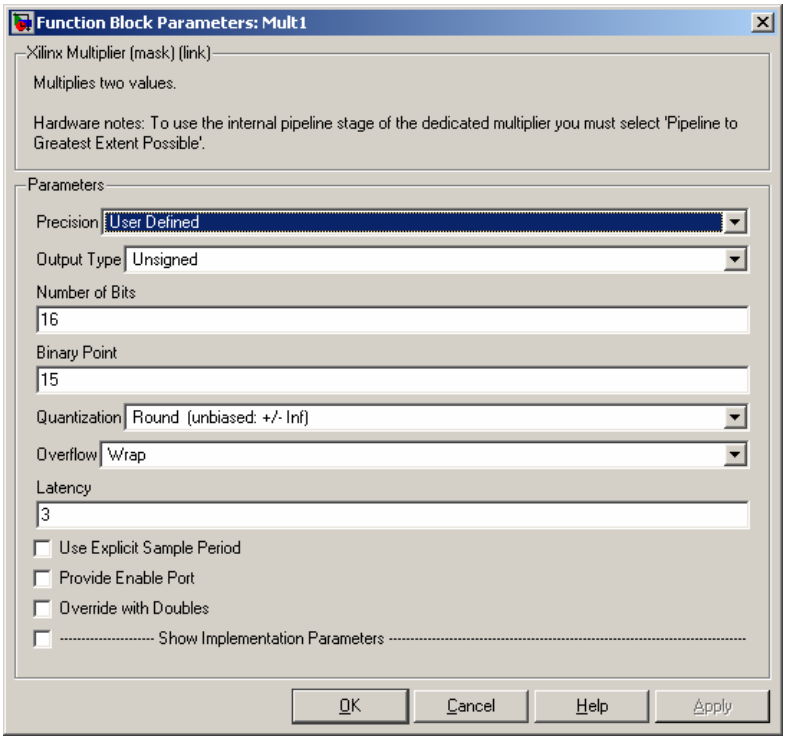

Figure C.1 Function Block Parameters of a Xilinx Multiplier Block

As seen in Figure C.1, FPGA provides flexibility in defining arbitrary fixed point arithmetic precision throughout the design and computation. The amount of FPGA hardware resources are directly related to the data width that is defined throughout design. Therefore it is wise to request only the data width and precision needed. Otherwise the FPGA resources are used excessively. This will cost more money, less circuit speed, and much power dissipation.

System Generator is particularly useful for algorithm exploration, design prototyping and model analysis. When these are the goals, the tool is used to flesh out an algorithm in order to get a feel for the design problems that are likely to be faced,

and perhaps to estimate the cost and performance of an implementation in hardware. Work is preparatory, and there is little need to translate the design into hardware. In this setting, a designer assembles key portions of the design without worrying about fine points or detailed implementation. Simulink blocks and MATLAB .m code provide stimuli for simulations, and for analyzing results. Resource estimation gives a rough idea of the cost of the design in hardware. Experiments using hardware generation can suggest the hardware speeds that are possible.