DEVELOPMENT OF AN FPGA BASED WAVELET TRANSFORMER FOR THE COMPRESSION OF RAW SAR DATA

by
KALEN B. BRUNHAM

An M.Sc. Thesis
Submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba, Canada

Thesis Advisor: W. Kinsner, Ph.D., P.Eng.

(xxxii + 210 + 1 + 14 + 17 + 16 + 31 + 30 =) 351 pages

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

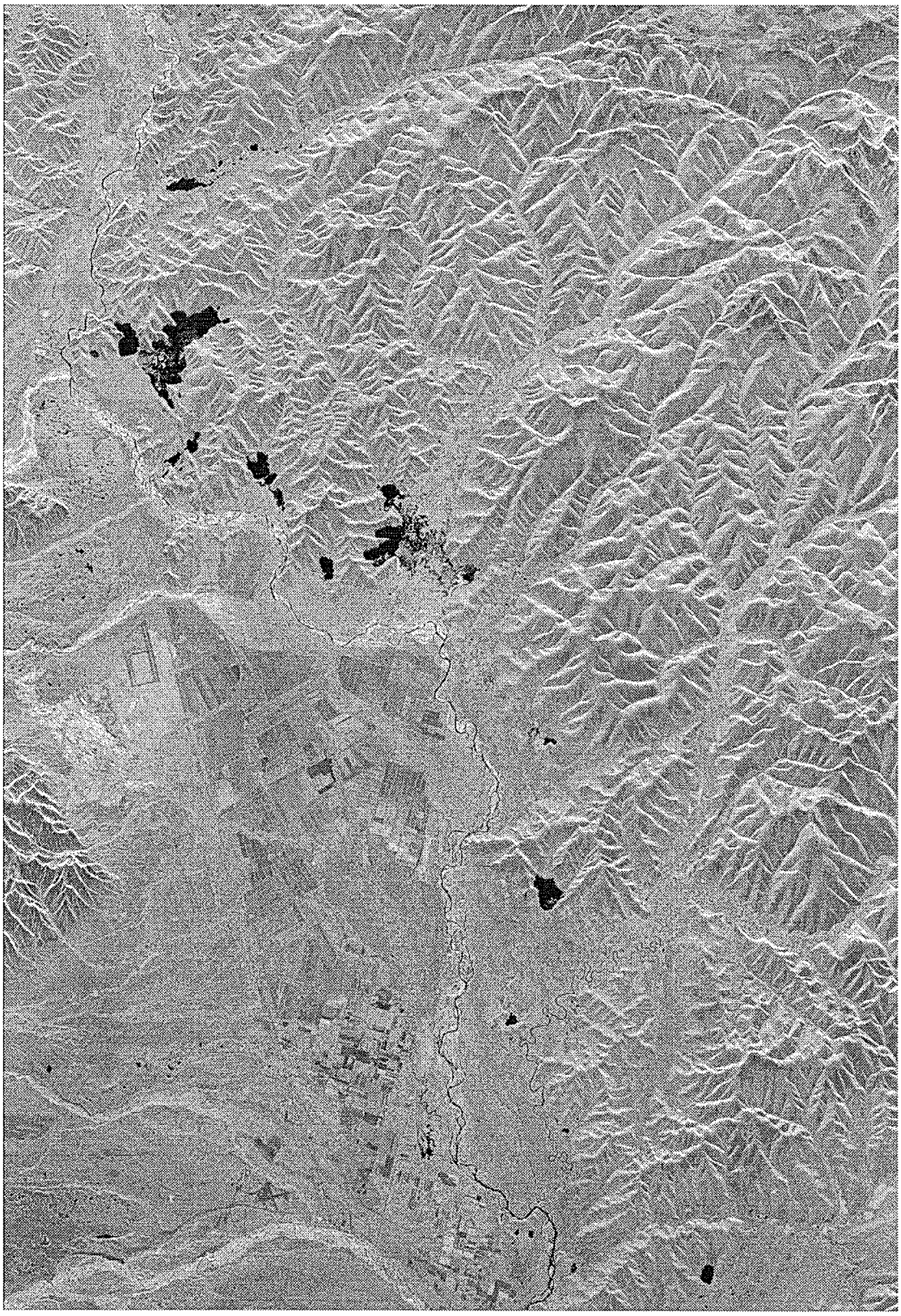
DEVELOPMENT OF AN FPGA BASED WAVELET TRANSFORMER FOR THE COMPRESSION OF RAW SAR DATA

BY

KALEN B. BRUNHAM

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree of

Master of Science



ABSTRACT

This thesis describes the design and implementation of the one-dimensional (1-D) discrete wavelet transform (DWT) using filter banks as a custom computing engine (CCE), using a Xilinx field programmable gate array (FPGA) for the compression of raw synthetic aperture radar (SAR) data.

The compression of raw SAR data has been a topic of interest to researchers for many years as the spaceborne SAR instrument generates an enormous amount of coherent information, which must be transmitted back to Earth for processing into the final SAR product. The goal of any new raw SAR data compression technique is to transmit the highest fidelity data from the radar satellite within the downlink bandwidth constraints. Since developing a theoretical compression technique is not a complete solution to the problem, the technique must be realizable in the hardware available for placement on the radar satellite. To this end, 2-D transform techniques are not practical due to the large on-board memory requirements. The implementation must not only be low in computational complexity, but must also be able to handle the ever increasing throughput demands of the radar.

Experimental results of the 1-D DWT technique on pre-conditioned raw SAR data in MatLab show that the best performing wavelet basis function is Daubechies-2, but that there is very little difference between all standard wavelet basis. The CCE implementation of the 1-D DWT technique gives an average SQNR of 8.95 dB on the five test sets which is 0.4 dB lower than that the average SQNR of BAQ.

ACKNOWLEDGEMENTS

Cette thèse a été rendue possible grâce à une allocation de recherche de Telecommunications Research Labs (TRlabs) et grâce à l'aide de nombreuses personnes. Je tiens à remercier mon directeur de thèse M. Witold Kinsner de son aide pour écrire et corriger cette thèse, et pour m'avoir encouragé à finir. J'aimerais aussi remercier le directeur de TRlabs Winnipeg, M. Jose Rueda, qui m'a aidé dans plusieurs situations et questions de recherche, ainsi que Mme. Michelle Harbin d'Alaska SAR Facility (ASF), pour nous avoir fourni les données brutes de SAR et leur processeur.

Pour ma recherche, j'ai été vraiment chanceux de travailler avec un chercheur et mathématicien compétent, M. Hakim El Boustani. C'est avec son aide que j'ai appris les ondelettes et les modèles pour les données brutes de SAR. Je suis sûr que sans lui, cette thèse, et ma compréhension des sujets, aurait été beaucoup moins riche. Merci d'être mon ami, et mon "co-directeur" de thèse. Je profite de cette occasion pour remercier les autres chercheurs du laboratoire, M. Sharjeel Siddiqui, M. Robert Barry, M. Neil Gadhok, M. Michael Potter, M. Stehpeh Dueck, M. Bin Huang et Mme. Jin Chen. Merci pour leur support dans les moments difficiles et leur sympathie de tous les jours.

Enfin, c'est l'encouragement de mes parents et de mes amis qui m'a permis de finir sans perdre la tête, et je les remercie profondément. Pendant mes études, il y a eu toujours une personne pour me soutenir et me conseiller. Ensemble on partage les moments faciles, mais aussi les moments difficiles. Je veux exprimer du fond du coeur mes sentiments à ma copine, ma meilleure amie, et mon amour Shawna Curtis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

1-D	**O**ne-**d**imensional
A/D	**A**nalog to **d**igital (converter/conversion)
ASF	**A**laska **S**AR **f**acility
ASIC	**A**pplication **s**pecific **i**ntegrated **c**ircuit
ASP	**A**laska **S**AR **p**rocessor
BAQ	**B**lock **a**daptive **q**uantization
CCM	**C**ustom **c**omputing **m**achine
CWT	**C**ontinuous **w**avelet **t**ransform
db-*x*	**D**aubechies *x* wavelet
DCT	**D**iscrete **c**osine **t**ransform
DSP	**D**igital **s**ignal **p**rocessor
DWT	**D**iscrete **w**avelet **t**ransform
ERS	**E**arth **r**esource **s**atellite (now **E**uropean **r**emote **s**ensing satellite)
FBAQ	**F**lexible **b**lock **a**daptive **q**uantization
FFT	**F**ast **F**ourier **t**ransform
FM	**F**requency **m**odulation
FPBAQ	**F**loating **p**oint **b**lock **a**daptive **q**uantization
FPGA	**F**ield **p**rogrammable **g**ate **a**rray
IC	**I**ntegrated **c**ircuit
ISLR	**I**ntegrated **s**ide-**l**obe **r**atio
JPL	**J**et **P**ropulsion **L**aboratory

LBG	**L**inde **B**uzo **G**ray (algorithm)
MRA	**M**ulti**r**esolution **a**nalysis
MSE	**M**ean squared **e**rror
NISLR	**N**ormalized **i**ntegrated **s**ide-**l**obe **r**atio
PDF	**P**robability **d**ensity **f**unction
PRF	**P**ulse **r**epetition **f**requency
PSNQR	**P**eak **s**ignal to **q**uantization **n**oise **r**atio
Radar	**Ra**dio **d**etection **a**nd **r**anging
SAR	**S**ynthetic **a**perture **r**adar
SLRAR	**S**ide **l**ooking **r**eal **a**perture **r**adar
SQNR	**S**ignal to **q**uantization **n**oise **r**atio
VHDL	**V**ery high-speed integrated circuit **h**ardware **d**escription **l**anguage
WFT	**W**indowed **F**ourier **t**ransform

LIST OF SYMBOLS

Radar

δ_A	Azimuth resolution
δ_D	Resolution of Doppler frequency measurement
δ_{GPC}	Ground range resolution using pulse compression
δ_R	Slant range resolution
δ_{RPC}	Slant range resolution using pulse compression
τ_p	Time duration of radar pulse
θ_a	Horizontal radar beamwidth
θ_n	Incidence angle
θ_L	Radar beam look angle relative to vertical
θ_r	Vertical radar beamwidth
θ_S	Radar beam squint angle relative to broadside
ϕ_k	Elementary phasor phase delay
λ_R	Radar beam wavelength
a_k	Elementary phasor reflectance amplitude
A_A	Antenna area
A_H	Antenna height

A_L	Antenna length
B_R	Frequency bandwidth of transmitted pulse
c	Speed of light in vacuum
H	Altitude of radar platform
f_D	Doppler frequency
f_{prf}	Pulse repetition frequency
f_r	Radar pulse frequency
f_s	Sampling frequency
t_{ipp}	Inter-pulse period
t_{prf}	Pulse repetition period
t_R	Round trip pulse time
R	Distance to target in range
R_f	Far range of radar beam
R_g	Distance to target in ground range
R_m	Distance in range to mid-swath
R_n	Near range of radar beam
R_s	Distance to target in slant range
V_c	Propagation speed of radar pulse

V_s	Speed of radar platform relative to the surface
W_a	Extent of radar beam footprint in azimuth
W_g	Ground range swath width
W_r	Slant range swath width

Compression

Σ	Symbol alphabet
θ_n	Transform coefficient
Δ	Quantization step
σ_i	Source symbol
σ_D^2	Mean squared quantization error
σ_x^2	Variance of x
$\sigma_{\theta_n}^2$	Variance of n th coefficient
B	Set of quantizer decision boundaries
b_i	Quantizer boundary
D	Distortion due to quantization
$D(x, y_i)$	Distortion metric for quantization of x to y_i
$E[x]$	Expectation of x
$f_x(x)$	Probability density function of x

H_1	First order entropy
M	Length of vector quantizer codebook
N	Length of vector quantizer codevectors
p_i	Probability of ith source symbol
$\boldsymbol{R}$	Set of all real numbers
R	Data rate
R_k	Number of bits assigned to kth coefficient
V_i	Quantization region of vector quantizer
X	Input codevector to vector quantizer
x	Source output
Y	Set of quantizer reconstruction values
y_i	Quantizer reconstruction value
$\boldsymbol{Z}$	Set of all integers

Wavelets

$\psi(t)$	Mother wavelet
$\Psi(\omega)$	Fourier transform of $\psi(t)$
$\phi(t)$	Scaling function
A_j	Approximation signal of $f[x]$ at resolution j

a_n^j	n th approximation coefficient at resolution j
D_j	Detail signal of $f[x]$ at resolution j
d_n^j	n th detail coefficient at resolution j
$L^2(\boldsymbol{R})$	Vector space of all square integrable real valued functions
V_j	Approximation space at resolution j
W_j	Detail space at resolution j
$w_{a,\,b}$	Wavelet coefficients

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

In the design of a spaceborne *synthetic aperture radar* (SAR) system, there exists a trade-off between the resolution of the resulting SAR image and the bandwidth allocated to the downlink channel. As radar technology advances and user demands for quality increase, current raw SAR data compression techniques will encounter serious limitations. To ensure that future imaging radars can convey the highest resolution possible, improvements in raw SAR data compression techniques are necessary. It must be noted, however, that the raw SAR compression techniques are very different from processed SAR image compression techniques as the statistics of raw SAR data are very similar to random white noise.

The basic spaceborne SAR system can be described in three distinct steps: (i) the transmission and reception of the radar pulses by the antenna, (ii) the downlink of the received radar echoes to Earth, and (iii) the processing of the received data into the final SAR product, such as a SAR image, as illustrated in Fig 1.1. This thesis is primarily concerned with improving the second step in the SAR system, namely downlinking as much information as possible within the bandwidth constraints.

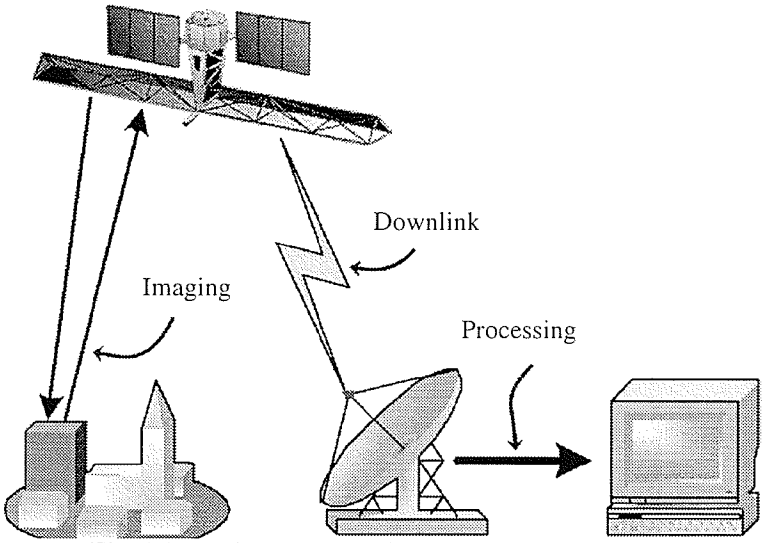


Fig. 1.1. Basic processes of a spaceborne SAR system.

The necessity for raw SAR data compression can be best understood by simply examining the Nyquist sampling frequency of the radar returns. If we consider the Canadian spaceborne SAR system, Radarsat-1, which has a minimum pulse bandwidth of B_R = 11.6 MHz, the Nyquist theorem then dictates that the minimum sampling frequency is 23.2 MHz, which results in an instantaneous bandwidth of 185.6 Mbps when quantized using an 8-bit *analog-to-digital* (A/D) converter. This data must then be transmitted in Radarsat-1's bandlimited channel of 105 Mbps, clearly compression is necessary. Currently, the most widely used raw SAR data compression technique is the *block adaptive quantization* (BAQ) [KwJo89] first developed by the NASA Jet Propulsion Laboratory (JPL). The BAQ algorithm is, however, not an optimal algorithm as it results in a low *signal to quantization noise ratio* (SQNR) compared to the theoretical Shannon bound. With new higher resolution

radars currently in development, compression techniques with low SQNR could preclude the advancements of SAR.

1.2 Objectives and Goals

The purpose of this thesis is to develop a *field programmable gate array* (FPGA) based *custom computing machine* (CCM) implementation of the 1-D *discrete wavelet transform* (DWT) for the compression of raw SAR data. The primary research questions addressed in this thesis are:

1. Can a 1-D DWT compression technique using a standard wavelet basis function be used in the compression of raw SAR data to achieve a SQNR greater than the SQNR of BAQ.
2. Which standard wavelet basis function gives the best SQNR in the compression of raw SAR data using the 1-D DWT compression technique.

It is not the aim of this thesis to satisfy the raw SAR data compression problem theoretically only, but also practically because a compression technique which requires a supercomputer, or massive on-board storage, is not a practical solution to the problem in the spaceborne environment. All *integrated circuits* (IC) for use in the space environment must be radiation hardened. Because of this, most electronics are implemented as custom *application specific integrated circuits* (ASIC) rather than using off-the-shelf devices such as *digital signal processors*. The FPGA allows the development an implementation that could be developed into an ASIC.

1.3 Organization of This Thesis

This thesis is organized into nine chapters and six appendices. Chapter 1 states motivation for this work and the research questions posed for this thesis.

Chapter 2 provides background information on radar, real aperture radar, and SAR systems. The main goal of this chapter is to introduce the SAR system and provide a discussion on the nature of the raw SAR signal.

Chapter 3 provides an introductory background on data compression and wavelets. The discussion on data compression refers to different compression techniques in order to provide a basis for discussion of current raw SAR data compression techniques. Later chapters in this thesis will apply the wavelet transform in the development of a raw SAR data compression technique.

Chapter 4 discusses current raw SAR data compression techniques and their resulting SQNR. This chapter demonstrates that currently, even the best techniques are either still quite far from the theoretical Shannon bound, or are prohibitively complex for implementation on a satellite.

Chapter 5 describes an application of the DWT for the compression of raw SAR data. In this chapter, several design parameters are discussed, including the choice of wavelet basis function. The wavelet basis design parameter is chosen in this chapter by investigating the SQNR of over 100 standard wavelet basis functions. Chapter 5 also

presents a discussion on pre-conditioning 4 and 5-bit per sample raw SAR data to 8-bit per sample data for experimental use.

In Chapter 6, the DWT compression technique developed in Chapters 5 is implemented as a CCE in a Xilinx FPGA. This chapter presents a discussion of architectural considerations and trade-offs in the design of the CCE. Implementation considerations for lowering the computation complexity of the CCE are also discussed. Experimental results of the FPGA based DWT CCM are provided in Chapter 7 along with implementation performance and a discussion of the results. Chapter 7 also presents a discussion of current metrics used to evaluate raw SAR data compression techniques.

Chapter 9 provides some concluding remarks and lists the contributions of this work. Recommendations for future work are also provided.

Appendix A of this thesis presents the characteristics and details of the Radarsat-1 and ERS-1 SAR satellites. Appendix B presents the details of the pre-conditioned SAR data test sets used in this thesis and shows the processed amplitude images of these test sets. Appendix C presents the MatLab code for the data pre-conditioning technique, and the BAQ and DWT compression techniques with the VHDL code for the DWT CCM listed in Appendix D. Additional figures and tables of results from Chapter 5 are shown in Appendix E with additional figures from Chapter 7 shown in Appendix F.

CHAPTER 2

BACKGROUND ON SYNTHETIC APERTURE RADAR

Synthetic aperture radar (SAR) is an all-weather imaging radar capable of high-resolution sensing in both the range and azimuth dimensions using the relative motion of the radar and the surface being imaged. In a SAR system, the radar instrument can be placed aboard either an airplane for airborne operation, or aboard a satellite for spaceborne operation. This thesis will concentrate on the case of the spaceborne radar as it is this type of SAR system which is most limited in bandwidth capacity for the downlink channel.

The spaceborne side-looking radar functions by emitting a high-frequency radar pulse directed at a surface below. The pulse is then reflected by the surface and received by the radar. The distance to objects on the surface can be ascertained by analysing the phase information of the reflected pulse. Information about the surface can also be obtained by examining the amplitude information of this reflected pulse, known as the backscattered signal. Because both amplitude and phase information are used, the raw SAR signal is a coherent signal.

The precision of a radar instrument is characterized by its spacial resolution [Tomi78]. This resolution determines the minimum separation of two point targets being sensed that can be distinguished as separate by the system. To obtain fine

resolution in the range direction, the classical technique of pulse compression [Cook60] is typically used. In real aperture radar, the azimuth resolution is proportional to λ_R / A_L, where λ_R is the wavelength of the imaging beam and A_L is the antenna length. For an imaging radar with a wavelength in the gigahertz (GHz) range, an antenna length of 1000 times larger than those used for terahertz (THz) optical frequencies is necessary to achieve the same resolution.

To improve the azimuth resolution without building an enormous antenna SAR is used. The SAR instrument is a standard radar that synthesizes a very large antenna by using the coherent sum of many radar echoes from a target. This allows SAR to achieve an azimuth resolution comparable to the range resolution and independent of the imaging wavelength used.

This chapter presents the necessary theory and practical concepts on SAR for the rest of the thesis. Using the background on the side-looking radar presented in the first half of this chapter, a model for the raw SAR data will be developed. Complete derivations of the equations used in this chapter can be found in [Brow67], [CuMc91 Ch.1], [Kova76 pp.21-31], [Tomi78], [Skol70], and [Lebe95].

2.1 Side-Looking Radar

2.1.1 Scanning Configuration

To image a surface below, the *side-looking real-aperture radar* (SLRAR) is carried on a moving platform at speed V_s relative to the surface in a straight line at a constant altitude H, as shown in Fig. 2.1.

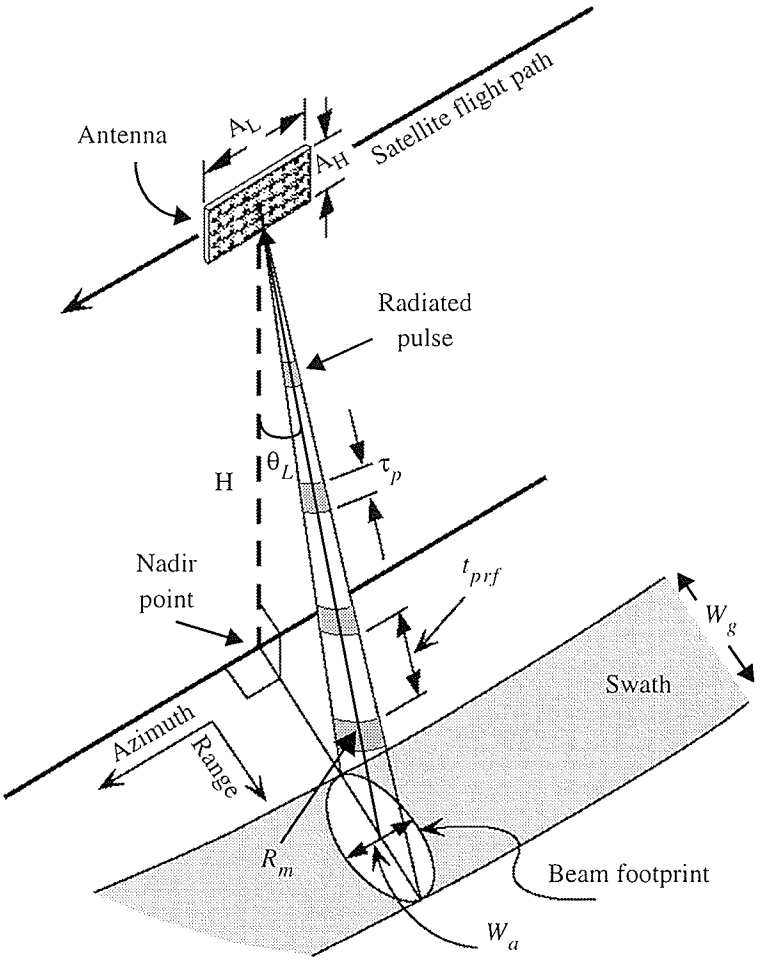


Fig. 2.1. Scanning configuration of a spaceborne SLRAR (after [Olms93]).

From this moving platform, the radar is directed with a squint angle θ_s relative to the flight path and downwards to the surface below with a look angle θ_L, relative to vertical. The most common squint configuration is to position the radar beam to be directed perpendicular to the flight path, making $\theta_s = 0°$. Using this geometry, the look angle θ_L is the same as the incidence angle θ_n, which is the angle between the radar beam and the normal to the Earth's surface at the point of interest.

The side-looking radar images a surface in two dimensions. The dimension perpendicular to the flight path of the radar is called the range dimension, and the dimension parallel to the flight path is called the cross-range or azimuth dimension.

The radar beam is wide in the vertical direction and so intersects the surface in an oval with the long axis extended in the range direction. The spread of the radar pulse in the azimuth direction, or the horizontal beamwidth θ_a of the radar, is determined by the antenna length A_L and the radar wavelength λ_R as

$$\theta_a = \frac{\lambda_R}{A_L} \tag{2.1}$$

The spread of the radar pulse in the range direction, or vertical beamwidth θ_r of the radar, is determined by the antenna width A_H and the radar wavelength as

$$\theta_r = \frac{\lambda_R}{A_H} \tag{2.2}$$

The range extent of the radar beam footprint on the ground W_g, called the ground swath width, and azimuth extent of the beam footprint W_a can be approximated as

$$W_g \approx \frac{\lambda_R H}{A_H \cos^2 \theta_L} \tag{2.3}$$

and

$$W_a \approx \frac{\lambda_R H}{A_L \cos \theta_L} \tag{2.4}$$

For the case of the radar aboard the Radarsat-1 satellite operating in standard mode with H = 800 km, $\theta_L = 20°$, A_H = 1.5 m, A_L = 15 m, and λ_R = 5.66 cm, the beam footprint is calculated to be $W_a \approx 32$ km and $W_g \approx 34$ km. More detailed characteristics of the Radarsat-1 satellite are given in Appendix A.

The side-looking radar is an active system, and can determine the distance from the radar antenna to targets on the surface below by transmitting pulses. Because the radar is looking at the target downwards at an angle θ_L, the distances between objects

are measured in a direction which is at a slant to the ground and are said to be recorded in slant range. In imaging applications, the separation in slant range is not as desirable as the real separation on the ground, known as the ground range, and can be obtained by dividing the slant range by the sine of the look angle. In the same manner, the ground range swath width W_g can be found from the slant range swath width W_r by

$$W_g = \frac{W_r}{\sin\theta_L} \tag{2.5}$$

A diagram of the slant range geometry is shown in Fig. 2.2.

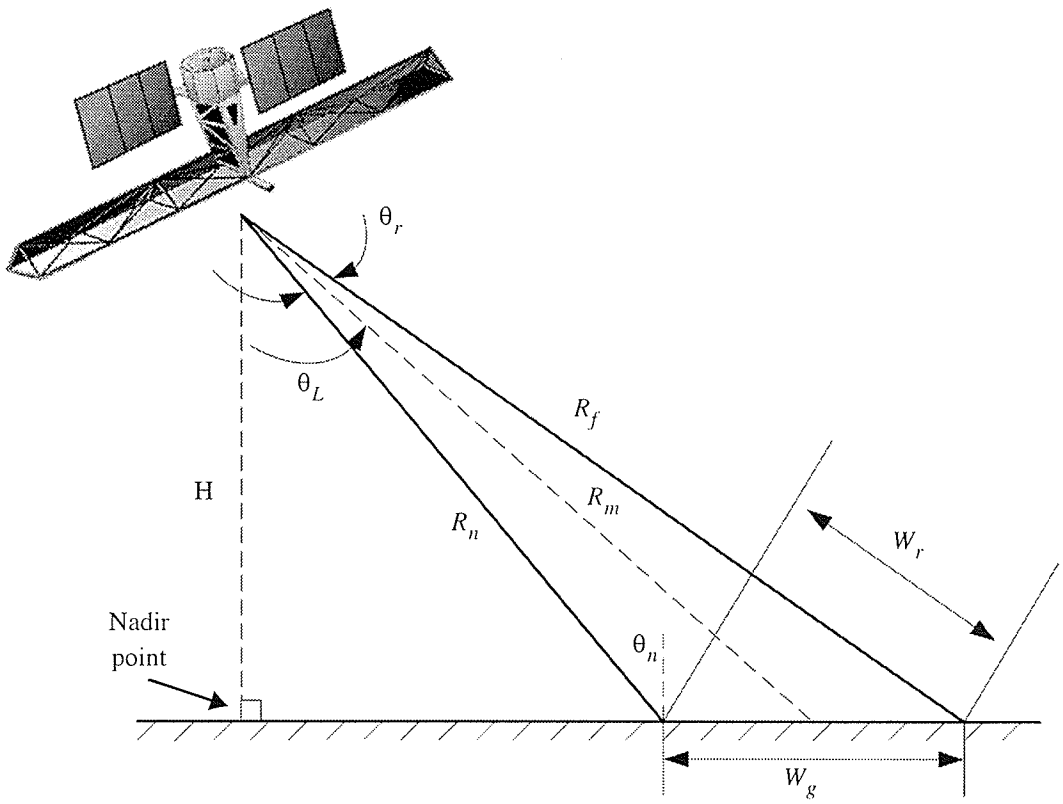


Fig. 2.2. Slant range geometry of a side looking radar (after [CuMc91]).

The radar geometry results in a radar beam footprint of length W_g in ground range. This geometry also defines the near range R_n and far range R_f which are the slant range to a target closest to the radar and farthest from the radar in the beam footprint.

2.1.2 Basic Acquisition Principle

While the radar is moving in space, it transmits a train of radar pulses at intervals determined by f_{prf}, the *pulse repetition frequency* (PRF). For each of these pulses radiated by the antenna, a pulse echo is received which provides information about the surface hit by the beam footprint. Between successive pulses, the radar platform moves a distance of V_s/f_{prf}, which is often much less than the azimuth extent of the beam footprint W_a. Because of this small platform displacement, the point targets in the radar beam footprint are hit by many different pulses from the radar. For example, a point target in midswath of the Radarsat-1 satellite is illuminated by $(W_a f_{prf})/V_s \approx 5500$ pulses. This type of radar imaging is known as strip-map imaging [MuVi89] and is illustrated in Fig. 2.3.

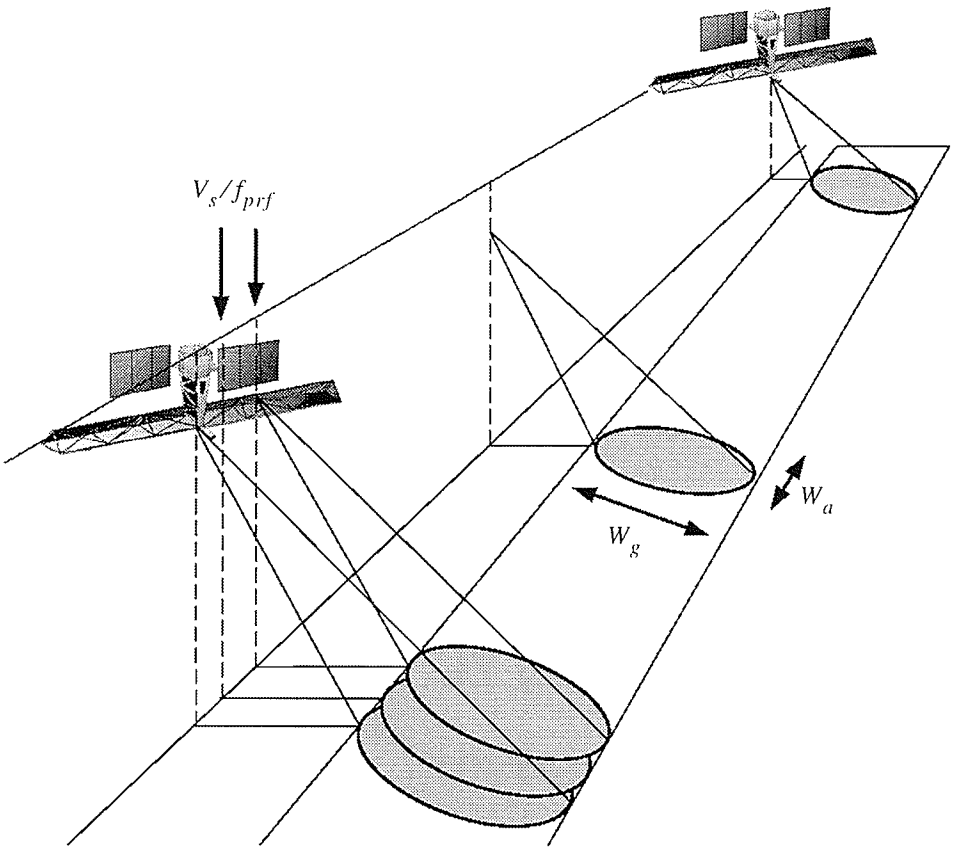


Fig. 2.3. Basic principle of strip-mapping SAR (after [Lebe95]).

Determining the appropriate PRF is a function of the geometry of the radar as after the radar pulse is radiated by the antenna, it goes into a listing mode to receive the echo. To avoid any echo overlap, the head of an echo from a point target in the near range R_n must arrive after the tail of the previous echo from a point target in the far range R_f separated by a inter-pulse period t_{ipp}. These times can be expressed in terms of distances as

$$\frac{2R_f}{V_c} < \frac{2R_n}{V_c} + t_{ipp} + \tau_p \tag{2.6}$$

where V_c is the propagation speed of the radar pulse and τ_p is the time duration of the radar pulse. Since the slant range swath width W_r is by definition the difference of the near range and far range, Eq. 2.6 can be written as

$$W_r = R_f - R_n < \frac{V_c t_{prf}}{2} \tag{2.7}$$

where $t_{prf} = t_{ipp} + \tau_p$. Equation 2.6 can then be combined with Eq. 2.3 to give t_{prf} in terms of the ground range swath width as

$$\frac{1}{t_{prf}} < \frac{W_g V_c}{2 \sin\theta_L} \tag{2.8}$$

Since the PRF f_{prf} is just the reciprocal of the $t_{ipp} + \tau_p$, Eq. 2.8 can then be combined with Eq. 2.2 to give t_{prf} as a function of the radar geometry

$$f_{prf} \leq \frac{1}{t_{prf}} = \frac{A_H \cos\theta_L V_c}{2\lambda_R H \sin\theta_L} = \frac{A_H f_R}{2H \tan\theta_L} \tag{2.9}$$

In the case of the radar aboard the Radarsat-1 satellite, the PRF is from 1270 Hz to 1390 Hz, which is well below the limit given by Eq. 2.7 of 4319 Hz to 13651 Hz for its range of look angles of 20° to 49°.

2.1.3 Radar Range Resolution

To detect the distance to an object, a pulse is generated from the transmitter aboard the radar platform, and radiated by the antenna as shown in Fig. 2.4.

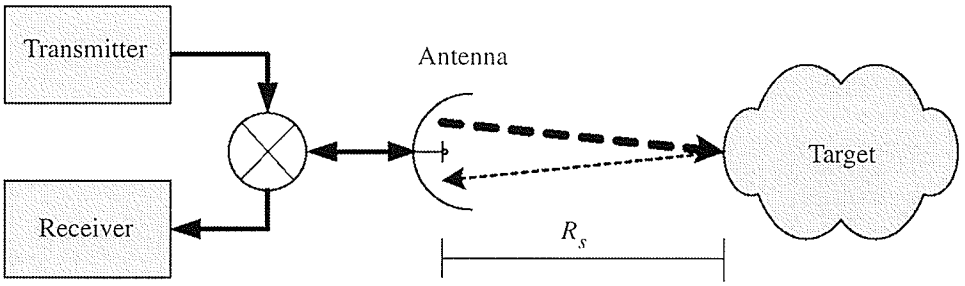


Fig. 2.4. Basic radar block diagram.

The greatly attenuated radar echo reflected from the target is then sensed by the receiver. The distance to the target, or range, is calculated using the round-trip time of the radar pulse t_R, and the propagation speed V_c of the electromagnetic wave, where V_c is typically taken to be c, the speed of light in a vacuum. The slant range to the target, which is the distance along which the radar signals propagate, is then given by

$$R_s = \frac{V_c t_R}{2} \tag{2.10}$$

The slant range resolution δ_R of the radar defines the minimum slant range separation of two points that can be distinguished as separate by the system. Intuitively, we can see from Eq. 2.10, that if the head of an echo pulse arrives at the

antenna at time $\tau > 0$ after the tail of the previous echo pulse, then they can be distinguished by the system as separate.

As an example, consider the system shown in Fig. 2.5 and the resulting received signal shown in Fig. 2.6. The radar radiates a pulse of duration τ_p towards the surface at $t = t_0$. At $t = t_1$, the radar pulse hits the point target A and an echo is reflected back towards the radar. At $t = t_2$ an echo is reflected from point target B and the radar pulse interacts with point targets C and D. At $t = t_3$, echoes from A and B are in transit and separated by a non-zero time, but the echoes from C and D are not distinguishable in time as the point targets were too close on the ground. The resulting received signal shown in Fig. 2.6. shows that A and B are distinguishable, but there is no clear distinction between where C ends and D begins.

The slant range resolution for the case of a single frequency pulse is therefore given by

$$\delta_R = \frac{V_c \tau_p}{2} \tag{2.11}$$

where τ_p is the time duration radar pulse.

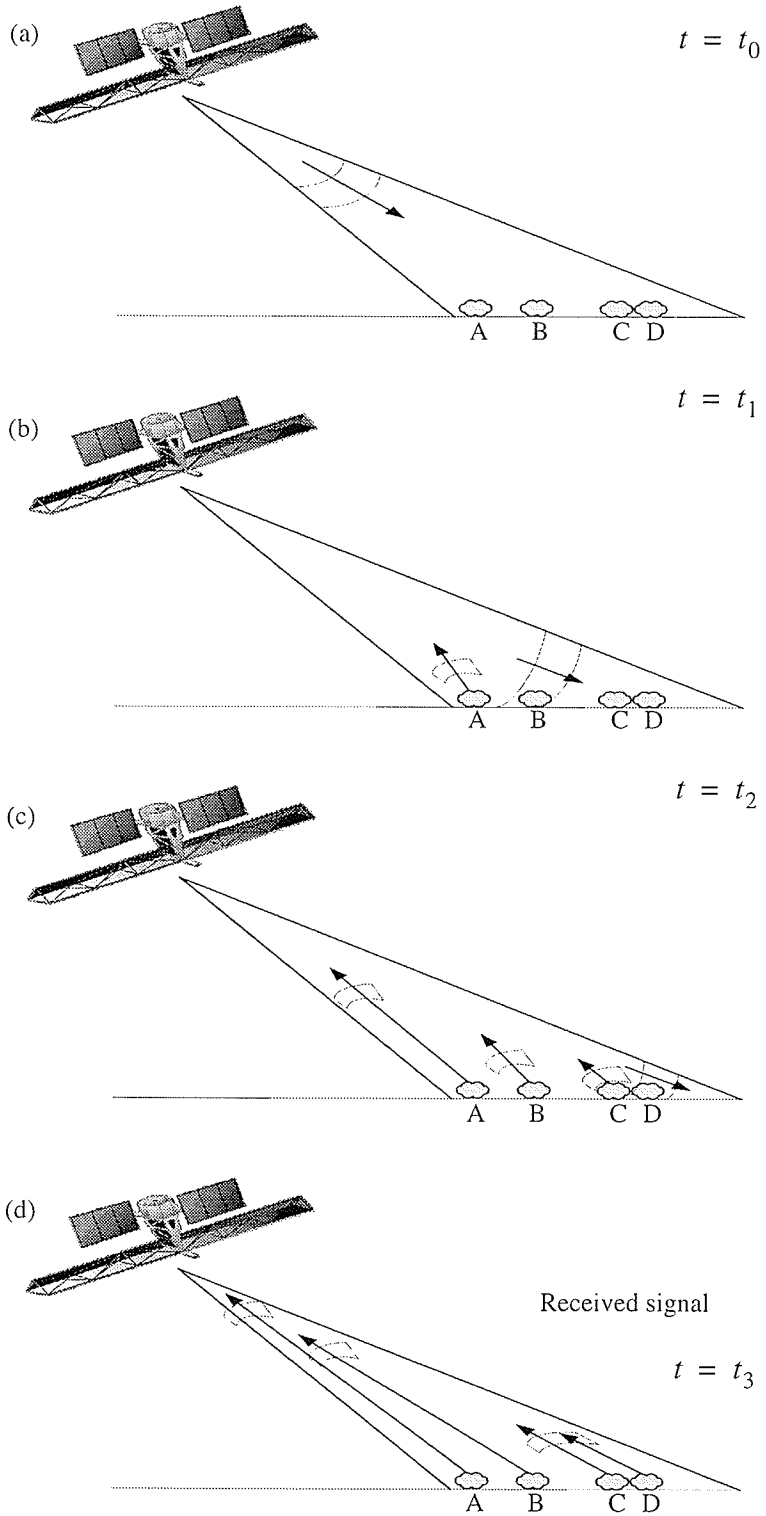


Fig. 2.5. Propagation of a radar pulse and separation of the echoes (after [Lebe95]). (a) The radar radiates a pulse. (b) The pulse hits the target A and an echo is reflected. (c) An echo is reflected from target B and the pulse interacts with targets C and D. (d) An echo from targets C and D is reflected.

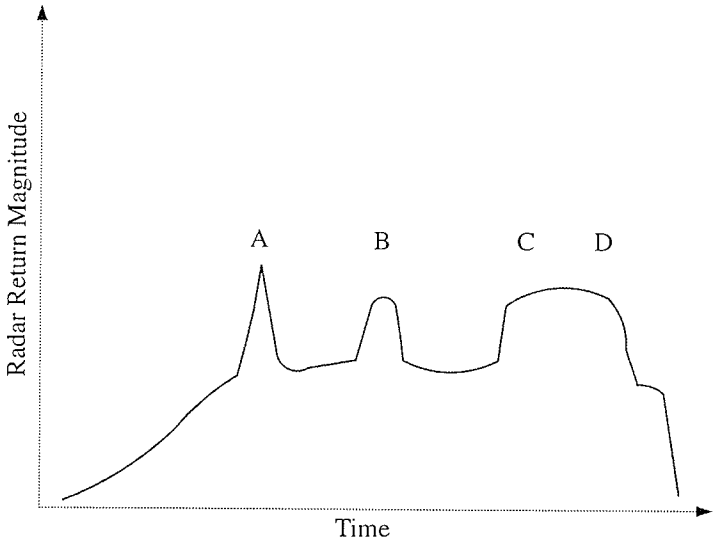


Fig. 2.6. Resulting received radar signal (after [Lebe95]).

Equation 2.11 suggests that reducing the time duration of the radar pulse will give improved range resolution indefinitely. However, as τ_p is reduced so is the energy contained within a pulse and this will eventually result in a pulse containing inadequate energy to produce a sufficient echo signal-to-noise ratio for reliable detection [CuMc91]. Increasing the pulse energy is not a complete solution either as the degree to which the radar pulse signal intensity can be increased is limited by the radar's available power and heat dissipation capabilities. To improve the range resolution, a radar pulse compression technique [Cook60] using a linear *frequency modulated* (FM) radar pulse is typically used where the frequency of the radar pulse changes linearly in time. A plot of a linear FM radar pulse is shown in Fig. 2.7.

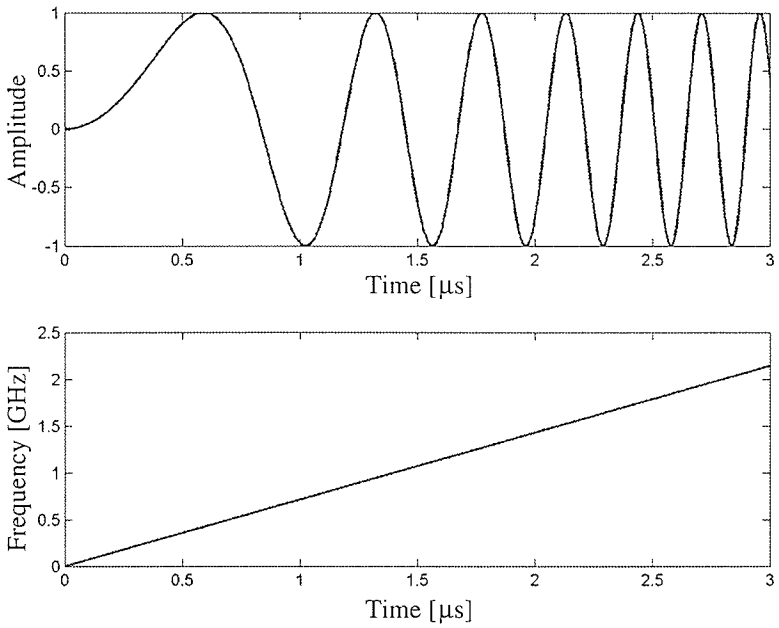


Fig. 2.7. Linear FM radar pulse.

One of the advantages of using a pulse compression technique is that even if the return from points at adjacent range intervals overlap in time, the shape of the pulse is distinctive enough for signal analysis to enable the components of the superimposed signals to be resolved using a matched filter [CuMc91], [Olms93]. This technique also permits the use of an extended pulse at lower intensity, with lower power requirements, which will still emit enough energy to give a detectable return. It can be shown [CuMc91] that the slant range resolution for a radar using this pulse compression technique is

$$\delta_{RPC} = \frac{V_c}{2B_R} \tag{2.12}$$

where B_R is the frequency bandwidth of the transmitted pulse. It can be further shown that this resolution can be made arbitrarily fine, within practical limits, by increasing the pulse bandwidth [CuMc91].

The ground range resolution, which is the resolution along the surface, can be obtained by dividing Eq. 2.12 by the sine of the look angle to give

$$\delta_{GPC} = \frac{\delta_{RPC}}{\sin\theta_L} = \frac{V_c}{2B_R \sin\theta_L} \tag{2.13}$$

Equation 2.13 shows that the radar system ground range resolution is determined by the geometry of the radar and the bandwidth of the radar pulse. In fact, all conventional radar systems, be it real aperture or SAR, resolve targets in the range direction in the same way. It is the resolution of targets in the azimuth direction that distinguishes SAR from other radar systems [CuMc91].

2.1.4 SLRAR Azimuth Resolution

The azimuth resolution of a SLRAR is defined as the minimum separation of two targets in azimuth that can be distinguished as separate by the system. Two such targets on the ground at the same slant range R_s can only be distinguished if they are

not both in the radar beam at the same time. Using Eq. 2.2, the azimuth resolution δ_{AR} at a slant range R_s is then given by

$$\delta_{AR} = R_s\theta_a = \frac{R_s\lambda_R}{A_L} \tag{2.14}$$

Equation 2.14 is very similar to the resolution in optics, and this equation demonstrates that the azimuth resolution of a SLRAR is proportional to the radar wavelength divided by the antenna length. This equation suggests that the azimuth resolution can be improved by simply increasing the antenna length. Unfortunately, this is not a possibility in all cases due to the mechanical problems involved in constructing a precise antenna with an A_L/λ_R ratio greater than a few hundred [CuMc91].

For example, if we wanted to obtain the Radarsat-1 azimuth resolution of 25 m using a wavelength of 5.66×10^{-3} m at a slant range of 850 km we would require an antenna length of about 2 km. Clearly, deploying an antenna this size in space is problematic at best, and an alternative is necessary.

2.2 Synthetic Aperture Radar

The goal of SAR is to achieve high resolution in both range and azimuth directions using a practically sized antenna. Its ground range resolution is determined by Eq. 2.5

where as the azimuth resolution is determined by the signal processing of the radar echoes.

The key observation that ultimately allowed for fine azimuth resolution with a manageable antenna length appears to have been the work of Carl Wiley of the Goodyear Aircraft Corporation in the early 1950s [ShRR62]. Wiley observed that a one-to-one correspondence exists between the azimuth coordinate of a reflecting object being linearly traversed by a radar beam, and the instantaneous Doppler shift of the signal reflected to the radar by that object. He concluded that a frequency analysis of the reflected signals could enable finer azimuth resolution than that permitted by the along-track width of the physical beam itself.

To better understand Wiley's observation, let us consider the radar echoes for an isolated point target as shown in Fig. 2.8. When the radar, travelling parallel to the point target with velocity V_s, reaches x_1, the point target is illuminated by the radar beam. This point target continues to be illuminated until the radar reach x_2, travelling a distance L. The slant range R_s to the point target is given by

$$R_s = \sqrt{R_0^2 + (x - x_0)^2} \tag{2.15}$$

where R_0 is the nearest range to the target, x is the position of the radar platform, and x_0 is the position of the radar platform when the range is R_0. The change in slant range ΔR_s of the target at any radar position x is then approximately

$$\Delta R_s \approx \frac{(x - x_0)^2}{2R_0} \tag{2.16}$$

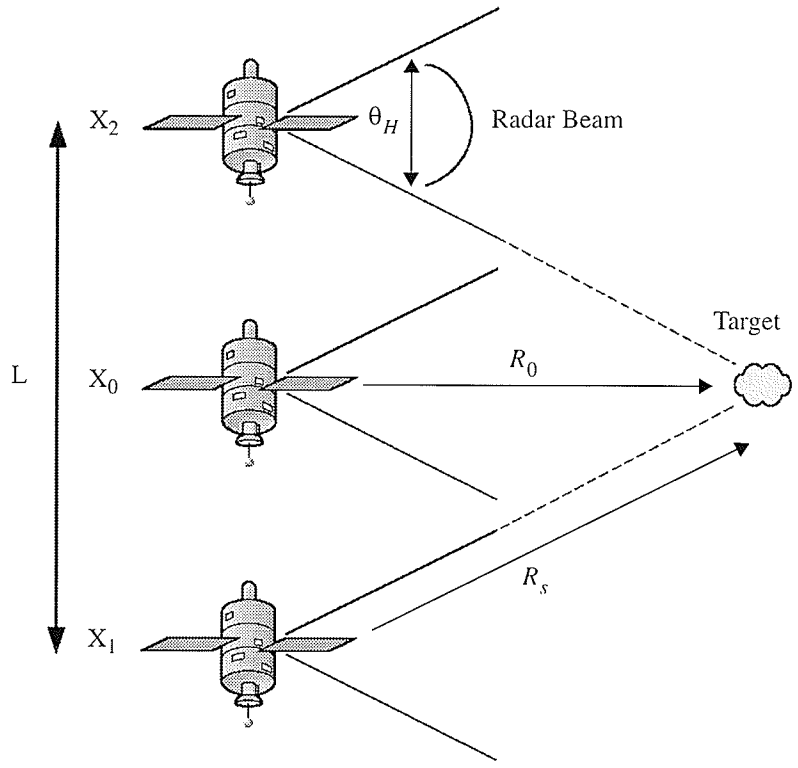


Fig. 2.8. Radar-target geometry (after [Tomi78]).

The phase change $\phi(x)$ corresponding to a change in slant range ΔR_s as a function of radar position x is given by the quadratic function

$$\Delta\phi = \frac{(x - x_0)^2}{\lambda_R R_0} \text{ [wavelengths]} \tag{2.17}$$

The phase change as a function of time can be obtained from Eq. 2.17 by assuming a constant velocity V_s for the radar platform and substituting the displacement $(x - x_0)$ with $V_s(t - t_0)$ to give

$$\phi(t) = \frac{V_s^2(t - t_0)^2}{\lambda_R R_0} \text{ [wavelengths]} \tag{2.18}$$

where the time variable t corresponds to the time at radar position x. As a time rate of change in Eq. 2.18 causes a frequency shift, we can calculate the Doppler frequency f_D by taking the first derivative of Eq. 2.18 to give

$$f_D = \frac{2V_s^2(t - t_0)}{\lambda_R R_0} \tag{2.19}$$

Therefore, if the received signal is frequency analyzed at time t_1, any energy observed in the return at a time t corresponding to a slant range R_s and at Doppler frequency

f_{D_1} will be associated with a target at coordinate $x_1 = \frac{\lambda_R R_s f_{D_1}}{2V_S}$. Similarly, energy at a different Doppler frequency f_{D_2} will be assigned to a corresponding coordinate x_2. This allows targets to be discriminated even though they are in the radar beam at the same time.

With the use of Doppler analysis of radar returns, the azimuth resolution δ_A is now related to the resolution of the measurement of the Doppler frequency δ_{f_D}. From Eq. 2.19 the azimuth resolution of a SAR is then

$$\delta_A = \frac{\lambda_R R_s \delta_{fD}}{2V_S} \tag{2.20}$$

which in theory results in

$$\delta_A = \frac{A_L}{2} \tag{2.21}$$

as shown by Cutrona *et al.* [CVLH61], [CuHa62]. Equation 2.21 presents the counter-intuitive result that arbitrarily fine azimuth resolution is obtainable by reducing the antenna length. This, however, assumes that the synthetic antenna length is equal to the distance across the radar beam at each range line, which is not always valid [CuMc91].

As a lower bound to this simple model the PRF equation from Eq. 2.9 can be examined to show [CuMc91]

$$W_s < \frac{c}{2f_p} < \left(\frac{c}{2V_S}\right)\delta_A \tag{2.22}$$

This equation shows that the swath width W_s decreases as the azimuth resolution is increased. By combining this result with Eq. 2.22, a requirement on the antenna area A_A is found [CuMc91] so that

$$A_A = A_H A_L > \frac{4V_S\lambda_R R_m(\tan\theta_L)}{c} \tag{2.23}$$

where R_m is the slant range from radar to mid swath.

2.2.1 SAR Subsystem Components

A typical SAR can be broken up into its major subsystems, as shown in Fig. 2.9. The radar functions by first having the *radio frequency* (RF) electronics send pulse-compressed radar pulses and then receiving the backscattered echoes using the same SAR antenna. The RF electronics then downconvert to baseband and amplify the received signal before splitting the demodulated signal into its real (or *inphase,* I) and imaginary (or *quadrature*, Q) components for further processing by the digital electronics.

Since older SARs did not use digital electronics to prepare the signal for transmission or recording, all processing was done on the analog SAR signal. All conventional SARs now digitize the I and Q signals using high-speed A/D converters, passing the digitized SAR signal to the digital electronics subsystem.

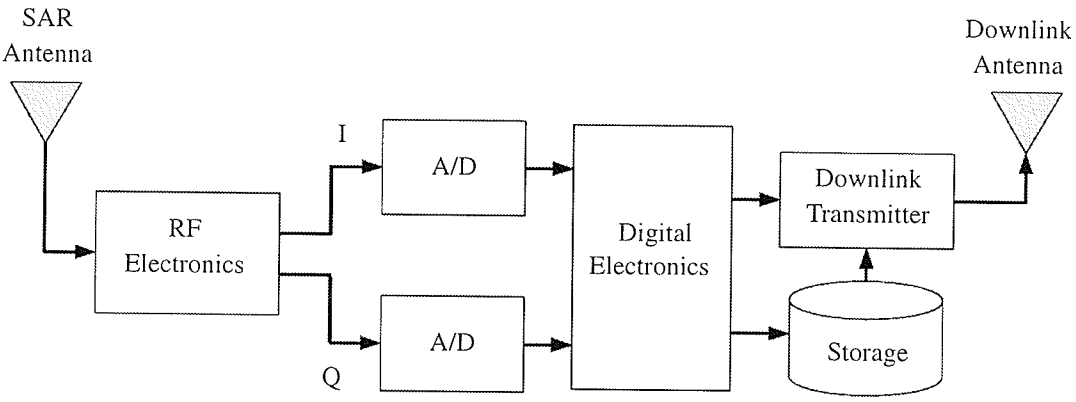


Fig. 2.9. Block diagram of major SAR subsystems.

The digitized SAR signal, called the raw SAR data, is then framed for on-board storage or transmission to the ground station. It is at this stage that compression of the raw SAR data can occur. In the event that the satellite is not in a position where it can communicate with the ground station, on-board storage is used for later transmission.

2.2.2 Nature of the SAR Signal

To develop a model for the received SAR signal, the statistics used to examine the properties of laser speckle patterns developed by [Good75], can be used. The backscatter returns from a synthetic aperture radar instrument can be modelled as the superposition of many small scatters within the antenna beam footprint [KwJo89]. The

coherent radar return $A(x,y,z)$ at observation point (x,y,z), which is a complex-valued function of space, is then given by

$$A(x, y, z) = \sum_{k=1}^{N} |a_k| e^{i\phi_k} \tag{2.24}$$

where a_k is the reflectance amplitude, ϕ_k is the phase delay, and the sum is over all elementary phasor contributions, N [Good75]. To illustrate this, Fig. 2.10 shows a time scale waveform of 1000 samples from the E1-25224R test set (as described in Appendix B).

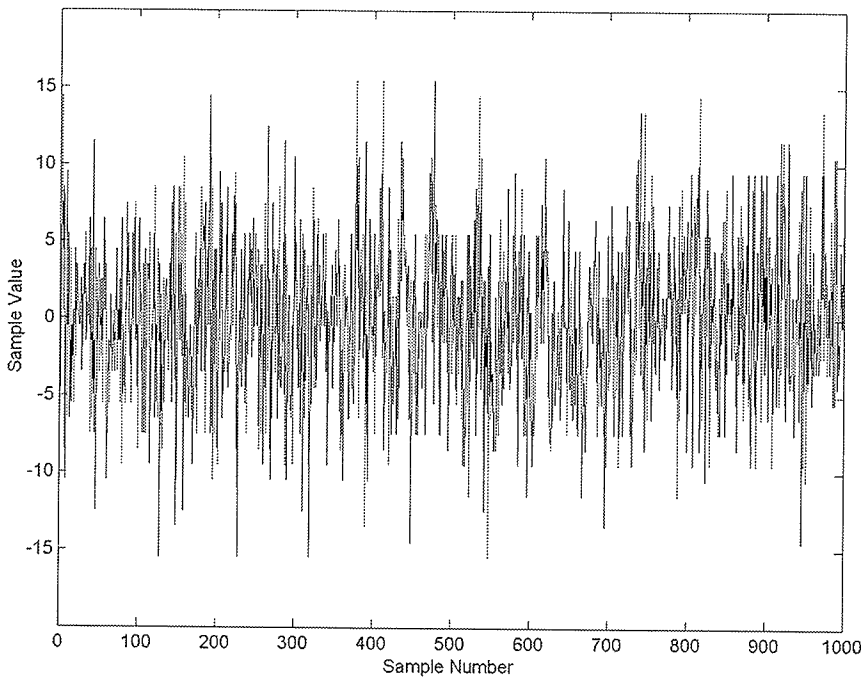


Fig. 2.10. Time waveform of E1-25224R test data.

Since the calculation of the signal statistics at point (x,y,z) is the statistical problem of a random walk in the complex plane, the randomly phased phasors can be assumed to have the following assumptions [Good75]:

(1) The amplitude a_k and phase ϕ_k of the kth elementary phasor are statistically independent of each other and of the amplitudes and phases of all other elementary phasors; and

(2) The phases ϕ_k are uniformly distributed on the interval $[-\pi,\pi]$.

Assumption (2) is a result of the scatters having an unknown range and the range resolution of the SAR being much greater than the wavelength of the transmitted SAR signal. This then produces the result that phase excursions of many times 2π radians produce a uniform distribution on the interval $[-\pi,\pi]$ [KwJo89]. These two properties allow the determination of the statistical characteristics of the real and imaginary parts, along with the magnitude and phase, of the received SAR signal.

The real and imaginary parts of the received SAR signal are given by

$$I = \text{Real}\{A\} = \sum_{k=1}^{N} |a_k| \cos\phi_k \tag{2.25}$$

$$Q = \text{Imag}\{A\} = \sum_{k=1}^{N} |a_k| \sin\phi_k \tag{2.26}$$

The expected values of the real and imaginary parts of the received SAR signal is given by

$$E[\mathrm{Re}\{A\}] = \sum_{k=1}^{N} E[|a_k|\cos\phi_k] = \sum_{k=1}^{N} E[|a_k|]E[\cos\phi_k] = 0 \qquad (2.27)$$

$$E[\mathrm{Im}\{A\}] = \sum_{k=1}^{N} E[|a_k|\sin\phi_k] = \sum_{k=1}^{N} E[|a_k|]E[\sin\phi_k] = 0 \qquad (2.28)$$

where Assumption (1) allows the calculation of the expected value of $|a_k|$ and ϕ_k separately, and Assumption (2) assures a value of zero for the expected values of both $\cos\phi_k$ and $\sin\phi_k$.

Proceeding in a similar fashion, the second order statistics of the real and imaginary parts of the received SAR signal are given by

$$\begin{aligned} E[\mathrm{Re}\{A\}^2] &= \frac{1}{N}\sum_{k=1}^{N}\sum_{m=1}^{N} E[|a_k||a_m|]E[\cos\phi_k\cos\phi_m] \\ &= \frac{1}{N}\sum_{k=1}^{N}\frac{E[|a_k|^2]}{2} \\ &= \sigma^2 \end{aligned} \qquad (2.29)$$

$$\begin{aligned} \mathrm{E}[\mathrm{Im}\{A\}^2] &= \frac{1}{N}\sum_{k=1}^{N}\sum_{m=1}^{N}\mathrm{E}[|a_k||a_m|]\mathrm{E}[\sin\phi_k\sin\phi_m] \\ &= \frac{1}{N}\sum_{k=1}^{N}\frac{\mathrm{E}[|a_k|^2]}{2} \\ &= \sigma^2 \end{aligned} \tag{2.30}$$

$$\begin{aligned} \mathrm{E}[\mathrm{Re}\{A\}\mathrm{Im}\{A\}] &= \frac{1}{N}\sum_{k,m=1}^{N}\mathrm{E}[|a_k||a_m|]\mathrm{E}[\cos\phi_k\sin\phi_m] \\ &= 0 \end{aligned} \tag{2.31}$$

where we have used the fact that independent and uniformly distributed phases exhibit the expectations

$$\mathrm{E}[\cos\phi_k\cos\phi_m] = \mathrm{E}[\sin\phi_k\sin\phi_m] = \begin{cases} \frac{1}{2} & k = m \\ 0 & k \neq m \end{cases} \tag{2.32}$$

$$\mathrm{E}[\cos\phi_k\sin\phi_m] = 0 \tag{2.33}$$

If we now suppose that the number N of elementary phasor contributions is very large, the central limit theorem dictates that the real and imaginary parts of the received SAR signal are Gaussian. Using this fact with the results from Eqs. 2.24 to

Eq. 2.16, the probability density function of the real and imaginary parts of the received SAR signal can be modeled as

$$f_X(X) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-X^2}{2\sigma^2}} \tag{2.34}$$

$$f_Y(Y) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-Y^2}{2\sigma^2}} \tag{2.35}$$

Thus, the real and imaginary parts of the received SAR signal can be modelled as uncorrelated Gaussian distributions with zero means and identical variances.

The magnitude M and phase θ of the received SAR signal can be related to the real and imaginary parts of A using the transformation

$$M = \mathrm{Re}\{A\}^2 + \mathrm{Im}\{A\}^2 \tag{2.36}$$

$$\theta = \arctan\frac{\mathrm{Im}\{A\}}{\mathrm{Re}\{A\}} \tag{2.37}$$

giving the probability density function of the magnitude M as

$$f_M(M) = \frac{1}{2\sigma^2} e^{\frac{-M}{2\sigma^2}} \tag{2.38}$$

and the probability density function of the phase θ as

$$f_\theta(\theta) = \frac{1}{2\pi} \tag{2.39}$$

Thus, the probability density function of the magnitude M of the received SAR signal can be modeled as a Rayleigh distribution and the phase as uniformly distributed on the interval $[-\pi,\pi]$. It should also be noted that

$$f_{M\theta}(M, \theta) = f_M(M) f_\theta(\theta) \tag{2.40}$$

and hence the magnitude and phase are statistically independent at any given point.

This model for the received SAR signal is very close to the true statistics of the received SAR signal. Figures 2.11 to 2.13 show histograms of a block of data from the E1-25224R test set, while Fig. 2.14 shows a normal probability plot of the same block of data. The large probability at the tails of the histogram can be attributed to the limited dynamic range of the A/D converter used to quantize the data aboard the ERS-1 satellite.

Figure 2.14 shows that the block of raw SAR data does in fact follow a normal distribution, except at the tails where there is deviation. Our model for raw SAR data also assumes that the samples are uncorrelated. If the signals were uncorrelated we would expect an autocorrelation plot as shown in Fig. 2.15.

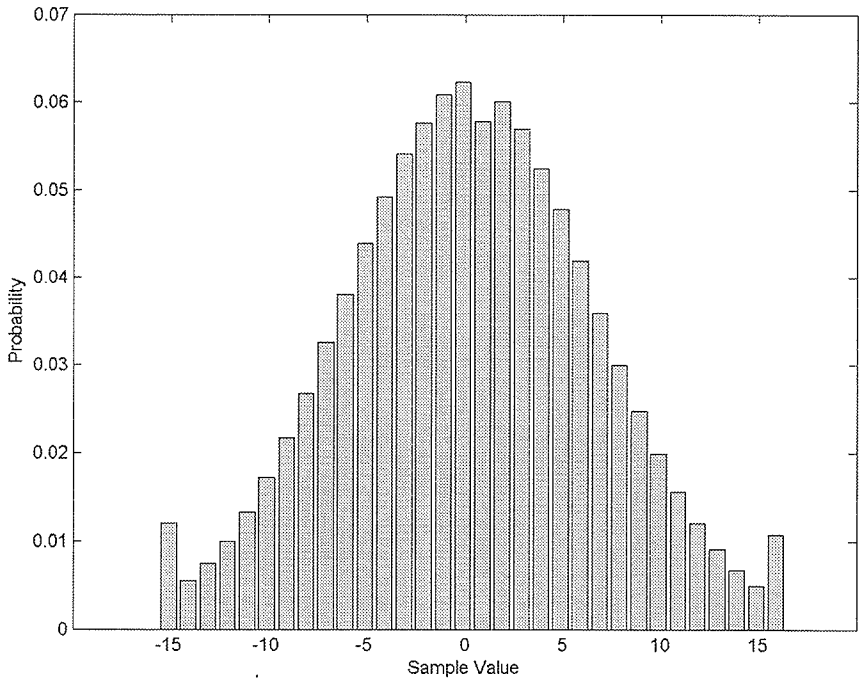


Fig. 2.11. Histogram of real and imaginary data from the ERS1-25224R SAR dataset.

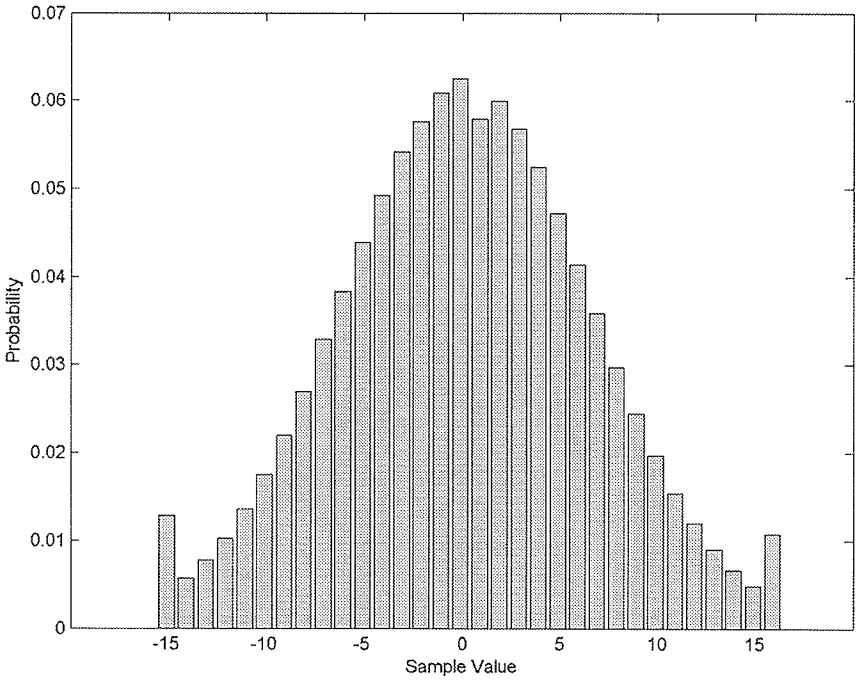


Fig. 2.12. Histogram of real data from the ERS1-25224R SAR dataset.

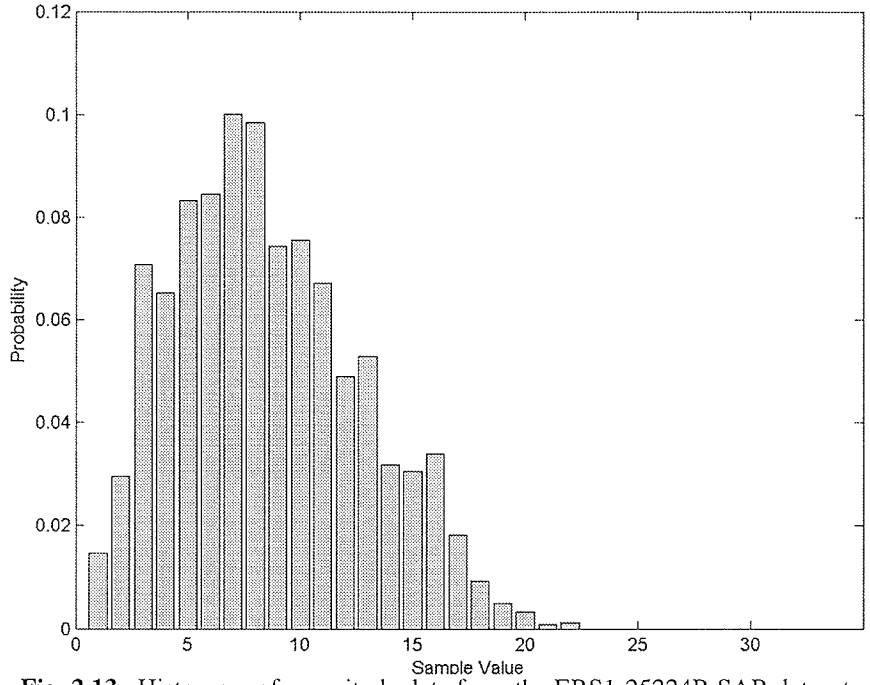


Fig. 2.13. Histogram of magnitude data from the ERS1-25224R SAR dataset.

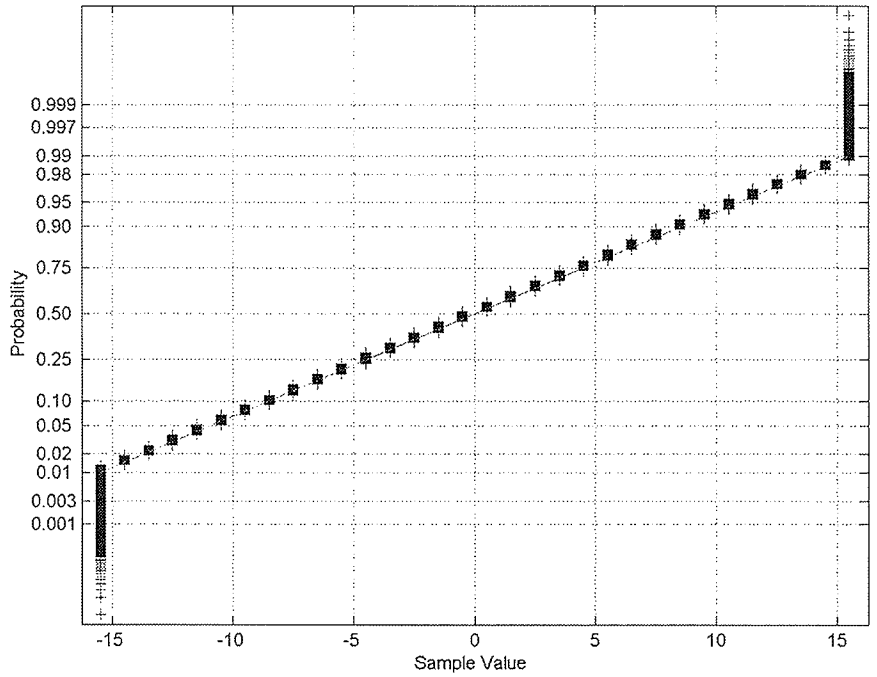


Fig. 2.14. Normal probability plot of ERS1-25224R raw SAR data.

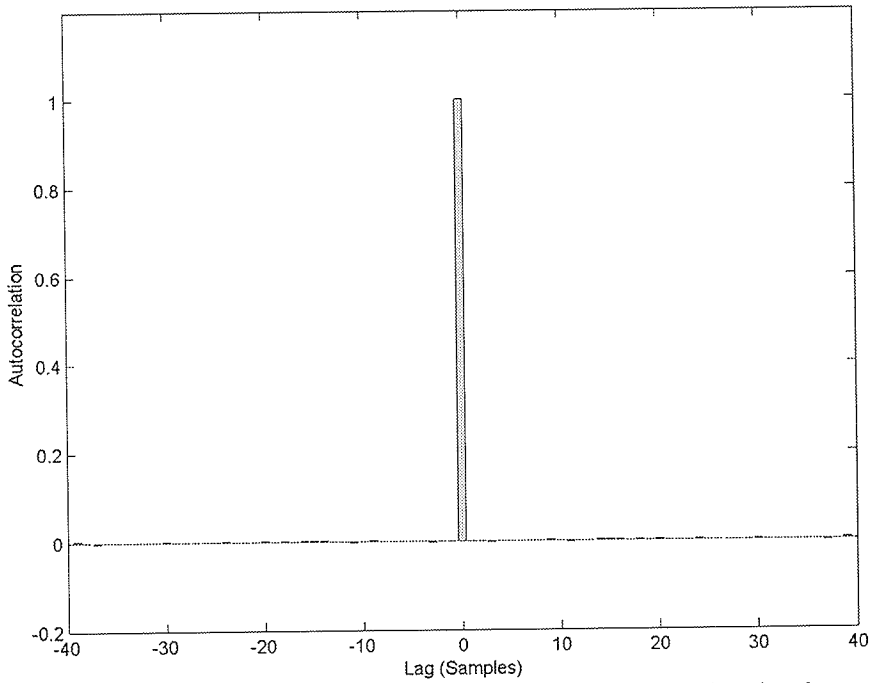


Fig. 2.15. Normalized autocorrelation plot of random Gaussian signal.

This plot shows that except for lag 0, which by definition is non zero, the random Gaussian signal has zero autocorrelation. When we now calculate the autocorrelation for successive range samples of ERS1-25224R raw SAR data, we see that the autocorrelation is no longer zero. Autocorrelation plots of the raw SAR data are shown in Figs. 2.16 and 2.17. It is this "low correlation" of the raw SAR data that will be the target of exploitation of the wavelet transform technique used in this thesis.

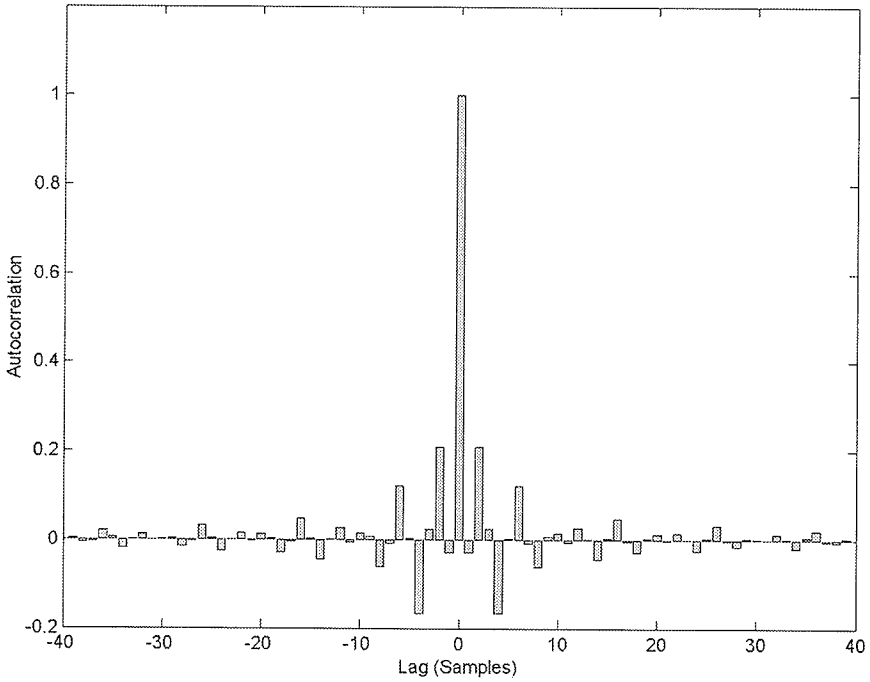


Fig. 2.16. Normalized autocorrelation plot of ERS1-25224R raw SAR data.

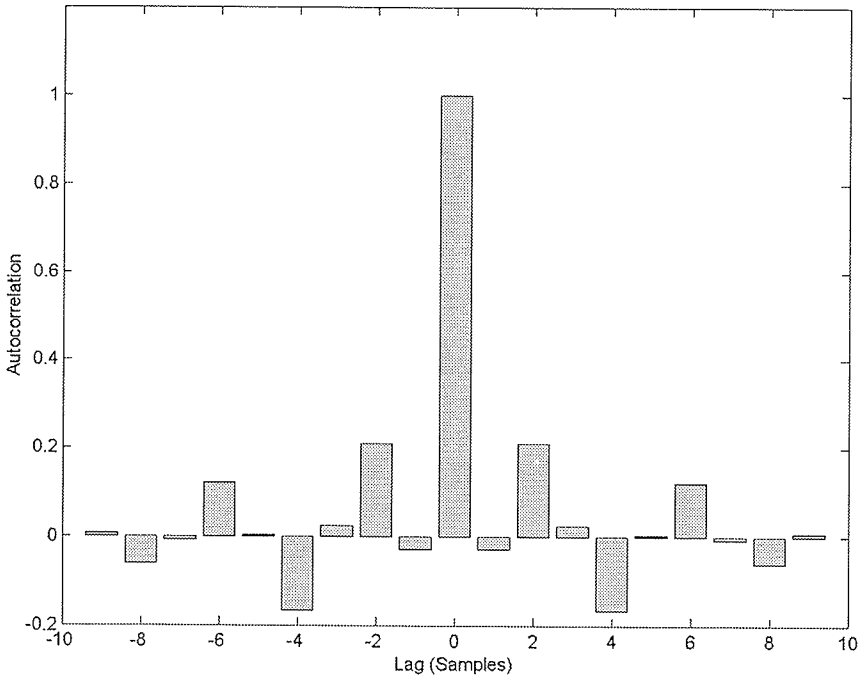


Fig. 2.17. Normalized autocorrelation plot of ERS1-25224R raw SAR data and lag of 10.

The data sets used in this thesis are from the ERS-1, ERS-2, and Radarsat-1 satellites, all of which operate at a single frequency and single polarization. Starting with the ENVISAT [MuCu95], which was launched on March 1, 2002, the SAR system on-board the spaceborne platforms will all be fully polarimetric SAR systems and will open a new generation in SAR technology. Although the model developed for the raw SAR signal does not address polarimetric data directly, the zero-mean Gaussian model presented in this section is still used to model polarimetric data [McCS98], [MuCu95].

2.2.3 SAR Processing

Once the raw SAR data have been received at the ground station it must be processed to form the final SAR product. It is this processing step that correlates all Doppler components for ground targets. Originally, this processing was done using optical lenses [CLPU66], however, all current processing is done using numerical computers. The conventional, and most widely seen SAR product is the SAR amplitude image, however, SAR systems are increasingly being used for non-imaging applications such as interferometric and multi-temporal differential applications. Today, diverse disciplines such as physical oceanography measuring moving currents, moving target detection on the ground, digital mapping, and global changes are common uses for SAR data.

The processor used in this thesis is the Alaska SAR Processor (ASP) from the Alaska SAR Facility (ASF) in Fairbanks Alaska. Details about the ASP can be found in [Olms93]. Processed images of all data sets used in this thesis are shown in Appendix B.

2.3 Summary

This chapter has discussed the basis of SLRAR and SAR systems and the azimuth resolution benefits of using SAR. This increased resolution is possible as the SAR systems employ the use of signal processing to detect multiple targets in the beam footprint. As a model for the raw SAR data, the laser speckle model developed by [Good75] has been suggested. This model shows that the statistics of the I and Q components of the raw SAR data are totally uncorrelated zero mean Gaussian, with identical variances. The zero mean Gaussian statistics conform with real SAR data, however, the data are not completely uncorrelated, as shown by plots of the range autocorrelation, rather raw SAR data is low-correlated. This model for the raw SAR data will be the foundation of the development of the DWT compression technique developed in the next chapters.

CHAPTER 3

BACKGROUND ON DATA COMPRESSION AND WAVELETS

The heart of compression is the notion of representing the information contained within a piece of data using a smaller number of information units. Since the reduction of information units has a limit, after which some information is actually lost, this type of compression is known as lossy compression, and is the class of compression techniques examined in this thesis.

To compress a piece of data (a sample) and reconstruct it using lossy compression, a process of removing information occurs. This processes is called quantization, and it is the process by which intervals of data values are represented by a single value. One of the most commonly used methods of quantization is the rounding of real numbers to their nearest integers. In such a quantization, the measure of the distortion caused by this loss can be evaluated by the mean of the square of the error of all data quantized, known as the mean squared error.

When performing scalar quantization on a string of data, the quantizer does not take advantage of any correlation that may exist within the string. As a result, the quantization may be suboptimal as the quantized data will still exhibit this correlation. One technique to remove the correlation, and enhance the compression by representing the data using fewer information units, is to apply a decorrelation

transform to the string of data. After the transform, there may be an energy compaction of transform coefficients which would allow a good representation of the original string using only a few of the coefficients. By quantizing only these important coefficients, the overall compression technique often provides less distortion. The wavelet transform is such a decorrelating transform and has been used for several year by many researchers.

This chapter presents the necessary theory and practical concepts on data compression and wavelets for the rest of the thesis. These techniques will be revisited in subsequent chapters as they are applied to the compression of raw SAR data. This chapter does not attempt to give a complete explanation for compression and the theory of wavelets, but rather provide a basis of relevant information for the reader. Additional information of compression can be found in [Sayoo96], [GeGr91], and [GrNe98], and readers interested in further resources for wavelets are directed to [Daub92], [Kais94], and [StNg96].

3.1 Data Compression

One of the key challenges for data compression can be simply explained by the problem of having too much information and not enough space (i.e., memory or bandwidth). A simple compression technique can be described as taking an input stream of symbols σ_j, where the set of symbols is called the alphabet, and transforming them into an output stream of codewords which utilize less space than

the input stream. This immediately leads to the question what exactly constitutes the information, and what is the impact of making the information fit in the space available. The definition of information used in this thesis is the definition by Shannon [Shan48] whereby the first order entropy of a memoryless source with symbol alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_N\}$ is given by

$$H_1 = -\sum_{j=1}^{N} p_j \log_2 p_j \qquad \text{[bits/symbol]} \tag{3.1}$$

where p_j is the probability of the jth symbol.

The entropy of a particular source tells us what the average amount of information would be for any message from that source. The entropy also tells us what is the maximum compression achievable without losing any information. If we take, for example, the case of 5-bit quantized raw SAR data from the E1-22089R data set, we find a 1st order entropy of 4.5452 bits. This tells us that there is only $5 - 4.5452 = 0.4548$ bits of redundancy in each of the 5-bit codewords. This entropy is an important threshold because if we attempt to fit this data into codewords of less than the entropy of the data then we will start to lose information. Compression of this type is known as lossy compression as we are losing some information.

3.2 Quantization

In many lossy compression applications, it is necessary to represent each source output using only a small number of codewords. Some information is lost as the number of possible distinct source output values is larger than the number of codewords available to represent them. Perhaps the most commonly used technique of quantization is the rounding off of real numbers to the nearest integer. More generally, a quantizer q can be defined to consist of a set of decision boundaries $B = \{b_i;\ i \in \boldsymbol{I}_1\}$ together with a set of reproduction values $Y = \{y_i;\ i \in \boldsymbol{I}_2\}$, where $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ are sets of consecutive integers. The overall quantizer q can then be defined as

$$q(x) = y_i \qquad x \in S_i \tag{3.2}$$

where $S_i = (b_{i-1}, b_i]$ is a partition of the real line S bounded by the decision boundaries b_{i-1} and b_i designed so that the partitions are disjoint and exhaustive. The error D due to the quantization q is for a single point x given as

$$D = x - q(x) = (x - y_i) \tag{3.3}$$

A simple quantizer with five reproduction levels is shown in Fig 3.1. This shows the quantizer as a collection of intervals bordered by the decision boundaries along with the reproduction value for each interval.

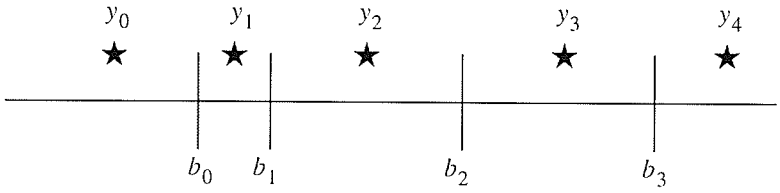


Fig. 3.1. A nonuniform quantizer with 5 reproduction levels.

A quantizer is said to be a uniform quantizer if the reproduction values y_i are equally spaced by an interval called the quantization step Δ and the decision boundaries b_i are midway between adjacent reproduction levels. If the quantization is to occur on the real line, and an infinite number of quantization levels are not available, all cells will have width Δ except for the two outermost cells which will be semi-infinite. An example of a uniform quantizer with a quantization step Δ and 8 reproduction values is shown in Fig. 3.2.

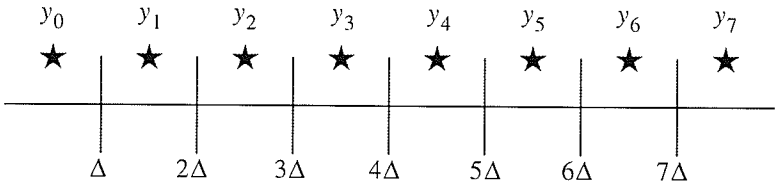


Fig. 3.2. A uniform quantizer with quantization step Δ and 8 reproduction values.

In compression the obvious goal of a quantizer is to minimize the error D so that the quantized data is close to the original. One way to determine the quality of the

quantizer is to define a distortion metric $D(x, y_i)$ to quantify the distortion resulting from the quantization of x to y_i. There have been many metrics developed to assess the distortion caused by a lossy compression technique, one of the most common is the *mean squared error* (MSE) defined as

$$\sigma_D^2 = \sum_{i=1}^{N} \int_{b_{i-1}}^{b_i} (x - y_i)^2 f_x(x) dx \tag{3.4}$$

where $f_x(x)$ is the *probability density function* (PDF) of the source x. To measure the size of the error relative to the signal, the ratio of the average squared value of the source output and the MSE can be calculated to give the *signal to quantization noise ratio* (SQNR) which is defined as

$$\text{SQNR} = 10\log_{10}\frac{\sigma_x^2}{\sigma_D^2} \quad [\text{dB}] \tag{3.5}$$

where σ_x^2 is the variance of the source x. Using the MSE as a metric, Shannon derived an equation to give the minimum achievable distortion of a Gaussian distributed source for a given rate R. From his rate-distortion theory, this distortion is given by

$$\sigma_D^2 = \sigma_x^2 2^{-2R} \tag{3.6}$$

Using Eq. 3.6, the SQNR can then be expressed as

$$\begin{aligned} SQNR &= 10\log_{10}\frac{\sigma_x^2}{\sigma_x^2 2^{-2R}} \\ &= 10\log_{10}\frac{1}{2^{-2R}} \\ &= 20\log_{10}2 \approx 6.02R \quad \text{[dB]} \end{aligned} \tag{3.7}$$

and is called the Shannon bound for a given rate R.

3.2.1 Scalar Quantization

Scalar quantization is the process of quantizing source strings, one symbol at a time. That is to say the quantizer will examine only one source output at a time, and then output the appropriate reconstruction value. There are two types of scalar quantizers, uniform and non-uniform quantizers. A popular example of uniform scalar quantizer is an A/D converter.

When using a non-uniform quantizer for a source with a known distribution it is optimum to use a PDF optimized quantizer. With the chosen distortion metric, the task of designing a PDF optimized quantizer is to choose a set of decision boundaries Y and reconstruction values B to minimize the distortion $\sigma_D^2 = f(B, Y)$.

Separately, Joel Max [Max60] and Stuart Lloyd [Lloy82] developed a set of equations to solve Y and B for a given PDF iteratively, using the MSE as a metric, giving rise to the Max-Lloyd algorithm as shown in Table 3.2. The SQNR of a Gaussian PDF optimized non-uniform quantizer is shown in Table 3.1 compared to the Shannon bound.

Table 3.1 SQNR of Gaussian optimized quantizer and Shannon Bound.

bits/sample	SQNR in dB	
	PDF Optimized	Shannon Bound
2	9.30	12.04
3	14.61	18.06
4	20.17	24.08

It can be shown that for a known probability distribution, the PDF optimized quantizer is the optimum scalar quantizer [Sayo96]. As can be seen from the SQNR results shown in Table 3.1, the PDF optimized quantizer is still quite far from the Shannon bound. In fact, the PDF optimized quantizer performs on average 4 dB worse than the Shannon bound for bit rates of 2-4 bits/sample. This disparity demonstrates that the optimal scalar quantizer is not the overall optimal quantizer, and that other quantization scheme may be superior.

Table 3.2 Max-Lloyd algorithm.

Step	Description
1	Start with a set of reconstruction values $\{y_i^{(0)}\}_{i=1}^{M}$ Set $k = 0$, $D^{(0)} = 0$. Select a threshold ε.
2	Find the decision boundaries $b_j^{(k)} = \dfrac{y_{j+1}^{(k)} + y_j^{(k)}}{2}$
3	Compute the distortion $D^{(k)} = \sum_{j=1}^{N} \int_{b_{j-1}^{(k)}}^{b_j^{(k)}} (x-y)^2 f_x(x)dx$
4	If $D^{(k)} - D^{(k-1)} < \varepsilon$ then stop, otherwise continue.
5	Set $k = k+1$. Compute the new reconstruction values $y_j^{(k)} = \dfrac{\int_{b_{j-1}^{(k-1)}}^{b_j^{(k-1)}} x f_x(x)dx}{\int_{b_{j-1}^{(k-1)}}^{b_j^{(k-1)}} f_x(x)dx}$ Go to step 2.

3.2.2 Vector Quantization

Vector quantization extends the concept of quantization from just one symbol at a time, to a string of symbols. The input string is quantized by comparing it to all codevectors in its codebook, to determine which one has the lowest distortion, then representing it by that codevector. This process is essentially a multi-dimensional quantization, where the decision boundaries of the quantizer now produce N-dimensional partitions of the N-dimensional space, where N is the length of the

codevectors. Vector quantization can be shown to always be at least as good as scalar quantization [GeGr91] because by considering more than one symbol at a time, the structure that may be present in the source can be exploited. The basic vector quantization algorithm is shown in Table 3.3.

Table 3.3 Basic vector quantization algorithm.

Step	Description
1	Create a N-dimensional vector X from N source samples.
2	Compute the distortion D between X and all codevectors $\{Y_i\}_{i=1}^{M}$
3	Represent the input vector X as the codevector with the smallest distortion.

The choice of codebook is of paramount concern when using vector quantization, as a poorly designed codebook will produce suboptimal results. Perhaps the most commonly used algorithm for codebook design is the one developed by Linde, Buzo, and Gray known as the LBG-algorithm [LiBG80]. Their key paper [LiBG80] generalized the Max-Lloyd algorithm for the case of a known source distribution where the inputs are no longer scalars, but are vectors. The generalized Max-Lloyd algorithm shown in Table 3.4.

Table 3.4 Generalized Max-Lloyd algorithm for codebook design.

Step	Description
1	Start with an initial set of reconstruction values $\{Y_i^{(0)}\}_{i=1}^{M}$ Set $k = 0$, $D^{(0)} = 0$. Select a threshold ε.
2	Find the quantization regions $V_i^{(k)} = \{X : d(X, Y_i) < d(X, Y_j) \quad \forall j \neq i\}$
3	Compute the distortion $D^{(k)} = \sum_{i=1}^{M} \int_{V_i^{(k)}} \left\| X - Y_i^{(k)} \right\|^2 f_x(X) dX$
4	If $\frac{D^{(k)} - D^{(k-1)}}{D^{(k)}} < \varepsilon$ then stop, otherwise continue.
5	Set $k = k + 1$. Find the new reconstruction values $\{Y_i^{(k)}\}_{i=1}^{M}$ that are the centroids of $\{V_i^{(k-1)}\}$. Go to Step 2.

An obvious practical problem with this algorithm is that it depends on knowing the multi-dimensional PDF of X. Furthermore integrals are required to calculate the distortion and centroids of odd-shaped regions in N-dimensional space at every iteration step.

Of greater practical interest is the codebook design algorithm developed by [LiBG80] for an unknown source distribution that works on a training set of vectors. The algorithm more commonly known as the LBG algorithm is shown in Table 3.5.

Table 3.5 LBG algorithm for vector quantizer codebook design.

Step	Description
1	Start with an initial set of reconstruction values $\{Y_i^{(0)}\}_{i=1}^{M}$ and a set of training vectors $\{X_n\}_{n=1}^{N}$. Set $k = 0$, $D^{(0)} = 0$. Select a threshold ε.
2	Find the quantization regions $\{V_i^{(k)}\}_{i=1}^{M}$ that are given by $V_i^{(k)} = \{X_n : d(X_n, Y_i) < d(X_n, Y_j) \ \ \forall j \neq i\} \qquad i = 1, 2, \ldots, M$
3	Compute the average of the distortions $D^{(k)}$ between the training vectors and their reconstruction value.
4	If $\frac{D^{(k)} - D^{(k-1)}}{D^{(k)}} < \varepsilon$ then stop, otherwise continue.
5	Set $k = k + 1$. Find the new reconstruction values $\{Y_i^{(k)}\}_{i=1}^{M}$ that are the average value of the elements in each of the quantization regions $V_i^{(k-1)}$. Go to Step 2.

The LBG algorithm guarantees that the distortion will not increase from iteration to iteration, but it does not guarantee that the algorithm will converge to the optimal solution. As can be seen from the algorithm steps, shown in Table 3.5, the final codebook is heavily dependent on the choice of the initial reconstruction values. To address this, [LiBG80] describe a codebook initialization for the LBG algorithm called the splitting technique. The splitting technique is shown in Table 3.6.

Table 3.6 LBG splitting technique.

Step	Description
1	Set $M = 1$ and define $Y_1^0 = \mathrm{E}[X]$, the centroid of the training set. Define a fixed purtubation vector ε.
2	Given the reconstruction values $\{Y_i^{(k)}\}_{i=1}^{M}$, "split" each vector Y_i into two close vectors $Y_i + \varepsilon$ and $Y_i - \varepsilon$. The set of reconstruction values now has $2M$ vectors. Replace $M = 2M$.
3	Apply the LBG algorithm on the reconstruction values $\{Y_i^{(k)}\}_{i=1}^{M}$ to obtain an M-level vector quantizer.
4	If $M = N$ then stop, otherwise go to step 2.

Using the splitting technique on a training set, the codebook is doubled at each iteration until the desired number of levels in the codebook is reached. By using the final codebook from the LBG algorithm of the previous iteration at each "splitting", the codebook is guaranteed to be at least as good as the codebook prior to splitting. If the desired number of levels is not a power of two, then, during the last iteration, instead of generating two new reconstruction values for each previous reconstruction value during the splitting step, only the necessary number of vectors are perturbed. The reconstruction points chosen to be perturbed should be those with the largest distortion, or the largest number of training set vectors.

3.3 Quantization in the Transform Domain

The motivation behind transform coding is that if a sequence of inputs is transformed into another sequence, in which most of the information is contained in only a few elements, then fewer bits would be required to represent the same information, or reduced distortion would be possible using the same number of bits.

For the case of a linear transform, the transform can be represented as

$$\theta_n = \sum_{i=0}^{N-1} x_i a_{n,i} \tag{3.8}$$

where a sequence of transform coefficients $\{\theta_n\}$ is obtained from the sequence of inputs $\{x_n\}$. The original sequence $\{x_n\}$ can be recovered from the transformed sequence via the inverse transform, which can be represented as

$$x_n = \sum_{i=0}^{N-1} \theta_i b_{n,i} \tag{3.9}$$

The characteristics of the transformed sequence are generally different than the characteristics of the input, as the characteristics of $\{\theta_n\}$ are determined by their position within the sequence. If the amount of information contained within the coefficients at different locations is different, it would be beneficial to assign more bits to the quantization of these coefficients. A measure of the differing information of the

different coefficients is the variance of each element $\sigma_{\theta_n}^2$, where $\sigma_{\theta_n}^2$ is the variance of the n^{th} coefficient. Using the MSE as the distortion criterion, a bit allocation for the transform coefficients can be found as [Sayo96]

$$R_k = R + \frac{1}{2}\log_2 \frac{\sigma_{\theta_k}^2}{\prod_{k=1}^{M} (\sigma_{\theta_k}^2)^{\frac{1}{M}}} \tag{3.10}$$

where R_k is the number of bits assigned to the k^{th} coefficient, R is the average bits per coefficient desired, and M is the number of coefficients to be quantized. This equation will minimize the R_ks, but does not guarantee that the bit allocations will be integers or even positive. The standard approach to this problem is to round all R_ks to the nearest integer, zero all negative R_ks, and then uniformly reduce all the non-negative R_ks until the average rate is equal to R.

The fundamental idea of transform coding is that by performing a suitable linear transformation on an input vector, a new vector can be obtained. The new vector, called the transform coefficients, might have the feature that the coefficients are much less correlated than the original samples. With this correlation removed, the information may be much more compact in the sense of being concentrated in only a

few of the transform coefficients. Having in this sense removed (or reduced) redundancy, the hope is to be able to quantize these coefficients more efficiently.

The objective is then to find the transform that most decorrelates the input. An optimal decorrelation for a Gaussian distribution was developed by Karhunen-Loeve, called the Karhunen-Loeve transform (KTL) [Sayoo96]. Unfortunately, KLT is impractical as it requires the calculation of the autocorrelation matrix for each block of input data [Sayoo96]. Consequently, other transforms, such as the wavelet transform, have recently become very attractive in transform coding due to their energy compaction capabilities, as described below.

3.4 Wavelets and the Wavelet Transform

3.4.1 Preliminaries

Before discussing the wavelet transform, it is useful to describe the vectorial space within which we will define the wavelet transform. The vector space used in this thesis is referred to as the $L^2(\boldsymbol{R})$ space and is the vector space of all square integrable real valued functions. A function $f(x)$ is said to be square integrable if

$$\int_x |f(x)|^2 dx < \infty \tag{3.11}$$

and any function that exists in the $L^2(\boldsymbol{R})$ space must satisfy Eq. 3.11.

3.4.2 Origins of the Wavelet Transform

Perhaps one of the most useful transformations of the 20th century in signal processing has been the Fourier transform, which decomposes a signal into sinusoidal waveforms of different frequencies. Users of the Fourier transform have been able to transform a signal mathematically from the time domain to the frequency domain.

Although for many applications, working in the frequency domain has been extremely useful and sufficient for the application, the Fourier transform has some major drawbacks. The most serious drawback is that all time information is lost in the Fourier transform. Without this time information, an event could be seen in the frequency domain, but no information regarding when it occurred would be available.

An advancement of the Fourier transform, to allow some time resolution, was developed by Gabor called the *windowed-Fourier transform* (WFT) [Daub92]. In the WFT, only a small "window" of the signal is analyzed at one time and is therefore able to provide some information about the time and the frequencies that occurred in the window. Unfortunately, due to the Heisenberg uncertainty principle, both time and frequency can not be known with absolute precision, and thus the WTF has limited precision for both time and frequency that is determined by the size of the window.

While the WFT does provide information about both frequency and time, once the window size is chosen for the time domain, that window size is fixed for all

frequencies. The wavelet transform addresses this problem of inflexible window sizes by examining the signal using variable sized regions. In wavelet analysis, long time regions are used where precise low-frequency information is required, and shorter time intervals where high-frequency information is required. Each of these regions allow the signal to be examined at different scales, and thus bypasses the problem of certainty outlined by Heisenberg. The window regions of the wavelet transform in contrast to the Fourier transform and SFT are shown in Fig 3.3.

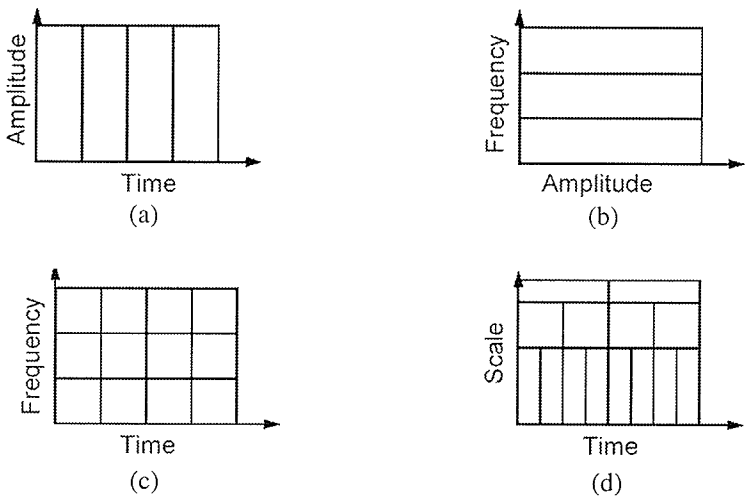


Fig. 3.3. Window shapes for (a) time domain, (b) frequency domain, windowed Fourier transform domain, and (d) wavelet domain (from [MMOP02]).

3.4.3 The Continuous Wavelet Transform

The wavelet transform can be used to decompose a signal $f(t)$ into wavelet coefficients using scaled and shifted versions of a single function as the basis for the

decomposition of a signal. This single function is known as the mother wavelet $\psi(t)$ and b-shifted and a-scaled versions are defined as

$$\Psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{3.12}$$

The wavelet coefficients $w_{a,b}$ with this basis function are obtained as the inner product of the original signal $f(t)$ with the $\psi_{a,b}(t)$ to give

$$w_{a,b} = \langle \psi_{a,b}(t), f(t) \rangle = \int_{-\infty}^{\infty} \psi_{a,b}(t) f(t) dt \tag{3.13}$$

The original function $f(t)$ can be recovered from the wavelet coefficients by

$$f(t) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} w_{a,b} \psi_{a,b}(t) \frac{da\,db}{a^2} \tag{3.14}$$

where

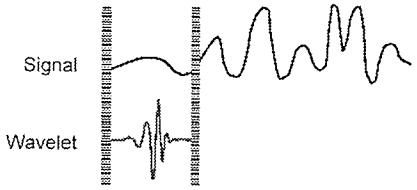
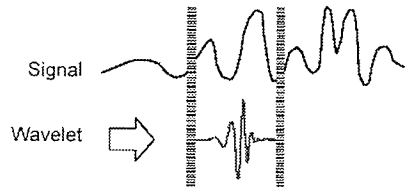
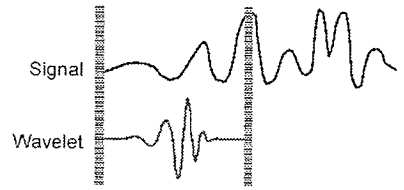
$$C_\psi = \int_0^{\infty} \frac{|\Psi(\omega)|^2}{\omega} d\omega \tag{3.15}$$

and $\Psi(\omega)$ is the Fourier transform of $\psi(t)$.

Intuitively, Eq. 3.14 holds true only when $0 < C_\psi < \infty$, which is known as the *admissibility condition*. For the admissibility condition to be satisfied it is necessary that $\Psi(0) = 0$, which says that the mother wavelet $\psi(t)$ must be zero mean (small wave). An additional requirement on $\psi(t)$ is that it belongs to $L^2(\boldsymbol{R})$. This implies that $|\Psi(\omega)|^2$ must decay to zero as ω goes to infinity. These requirements on $\psi(t)$ mean that the energy in $\Psi(\omega)$ is localized in a narrow frequency band which in turn allows the wavelet frequency localization capabilities. The wavelet transform algorithm is shown in Table 3.7.

When both the scaling parameter a and shifting parameter b are continuous, then the transform is said to be the *continuous wavelet transform* (CWT). Unfortunately, the CWT is not useful in practice as there is an infinite number of coefficients and each coefficient requires an integration over all time. To make the wavelet transform more practical, the scaling and shifting parameters can be discretized to give the *discrete wavelet transform* (DWT).

Table 3.7 Wavelet transform algorithm (from [MMOP02]).

Step	Description
1	Take a wavelet and compare it to a section at the start of the original signal.
2	Calculate the correlation $w_{a,b}$ between the wavelet is with this section of the signal.  $W_{a,b} = 0.122321$
3	Shift the wavelet to the right and repeat steps 1 and 2 until the whole signal has been covered.  $W_{a,b} = 0.8761$
4	Scale the wavelet and repeat steps 1 through 3.  $W_{a,b} = 0.7642$
5	Repeat steps 1 through 4 for all scales.

3.4.4 Discrete Wavelet Transform

Intuitively, the discrete versions of a and b must be related to each other because if the scale makes the basis function narrow, then the shifting steps should be correspondingly small, and vice-versa. There are a number of ways that a and b can be chosen, however, the most popular are

$$a = a_0^m \tag{3.16}$$

$$b = nb_0a_0^m \tag{3.17}$$

where both m and n are integers, $a_0 = 2$, and $b_0 = 1$. The wavelet basis functions of the DWT for an m-scale and n-shift are given by

$$\Psi_{m,n}(t) = 2^{-m/2}\psi(2^{-m}t - n) \tag{3.18}$$

with wavelet coefficients

$$w_{m,n} = \langle f(t), \Psi_{m,n}(t)\rangle = a_0^{-m/2}\int f(t)\psi(a_0^{-m}t - nb_0)dt \tag{3.19}$$

If the admissibility condition is satisfied, then reconstruction is possible from the wavelet coefficients by

$$f(t) = \sum_m \sum_n w_{m,n} \psi_{m,n}(t) \tag{3.20}$$

In the CWT, both the time function $f(t)$ and the transform are continuous. The DWT improves the practicality of the wavelet transform by discretizing the scaling and shifting parameters to give a wavelet series for the continuous time function $f(t)$. In the real world, signals of interest are usually sampled functions that are discrete in time. For the wavelet transform to be useful in the analysis of such signals, both the time and the frequency representation must be discrete. Several techniques exist to deal with such signals, the most common being the *multiresolution analysis* (MRA) developed by Mallat [Mall89].

3.4.5 Multiresolution Analysis

The general procedure behind Mallat's MRA [Mall89] is to decompose a discrete signal $f[x]$ into an approximation signal $A_j[x]$ and a detail signal $D_j[x]$, where j is the resolution of the multiresolution analysis, and the brackets signify the specific element of the discrete signal.

3.4.5.1 Mathematical Preliminaries

A multiresolution analysis in $L^2(\boldsymbol{R})$ consists of a sequence of successive approximation spaces V_j that satisfy the following properties:

(1) The spaces are nested subspaces of lower resolution spaces

$$v_\infty = \{0\} \subset \ldots \subset V_1 \subset V_0 \subset V_{-1} \subset \ldots \subset V_\infty = L^2 \tag{3.21}$$

as shown graphically in Fig. 3.4.

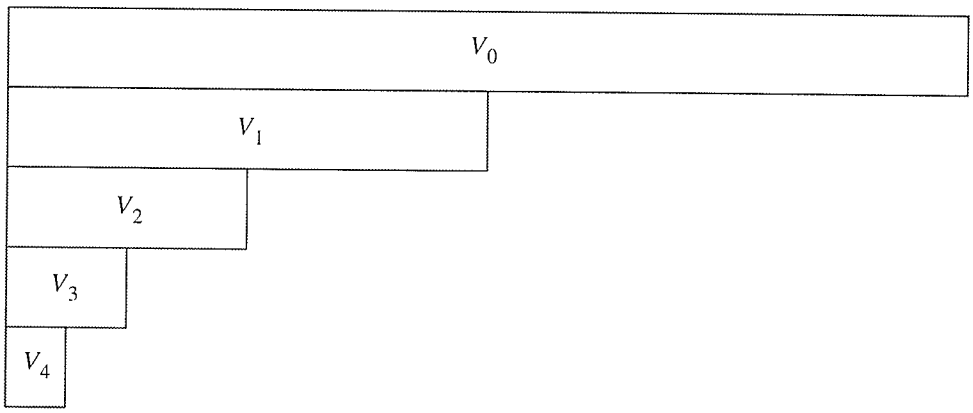


Fig. 3.4. MRA approximation spaces.

(2) The closure of the union of all subspaces must be the $L^2(\boldsymbol{R})$ space

$$\overline{\bigcup_{j \in Z} V_j} = L^2(\boldsymbol{R}) \tag{3.22}$$

(3) The intersection of all subspaces must be the null space

$$\bigcap_{j \in Z} V_j = \{0\} \tag{3.23}$$

(4) The space of a function is invariant to integer shifting

$$f[x] \in V_j \Leftrightarrow f[x-k] \in V_j \qquad \forall k \in \boldsymbol{Z} \tag{3.24}$$

(5) All spaces are scaled versions of the central space V_0

$$f(x) \in V_j \Leftrightarrow f(2^j x) \in V_0 \tag{3.25}$$

If the spaces V_j satisfies Properties 1 to 5, then the spaces V_j form an MRA and there exists a function $\phi(x) \in L^2(\boldsymbol{R})$ called the scaling function of the MRA where

$$\{\phi_{j,n}; n \in \boldsymbol{Z}\} \text{ is an orthonormal basis of } V_j \tag{3.26}$$

and $\phi(x)$ is defined for any 2^j-scaling and n-shifting as

$$\phi_{j,n}(x) = \left(\frac{1}{\sqrt{2}}\right)^j \phi(2^{-j}x - n) \tag{3.27}$$

and the projection of a signal $f[x]$ onto V_j is called an approximation of the signal $f[x]$ at the resolution j.

From the subspaces $(V_j, j \in \boldsymbol{Z})$ in Eq. 3.21, we define another sequence of subspaces of $L^2(\boldsymbol{R})$, $(W_j, j \in \boldsymbol{Z})$ where W_j is the orthogonal complement of V_j in V_{j-1} such that

$$V_{j-1} = V_j \oplus W_j \tag{3.28}$$

$$W_j \perp W_k \text{ for } j \neq k \tag{3.29}$$

$$\bigoplus_{j \in Z} W_j = L^2(\boldsymbol{R}) \tag{3.30}$$

as shown graphically with the spaces V_j in Fig. 3.5. It can be shown that under the MRA assumptions, there exists a function $\psi(x)$, called the mother wavelet, such that

$$W_j = Span\{\psi_{m,n}, n \in \boldsymbol{Z}\} \tag{3.31}$$

and $\{\psi_{j,n}; j, n \in \boldsymbol{Z}\}$ forms an orthornormal basis of $L^2(\boldsymbol{R})$. The function $\psi(x)$ is defined for any 2^j-scaling and n-shifting as

$$\psi_{j,n}(x) = \left(\frac{1}{\sqrt{2}}\right)^j \psi(2^{-j}x - n) \tag{3.32}$$

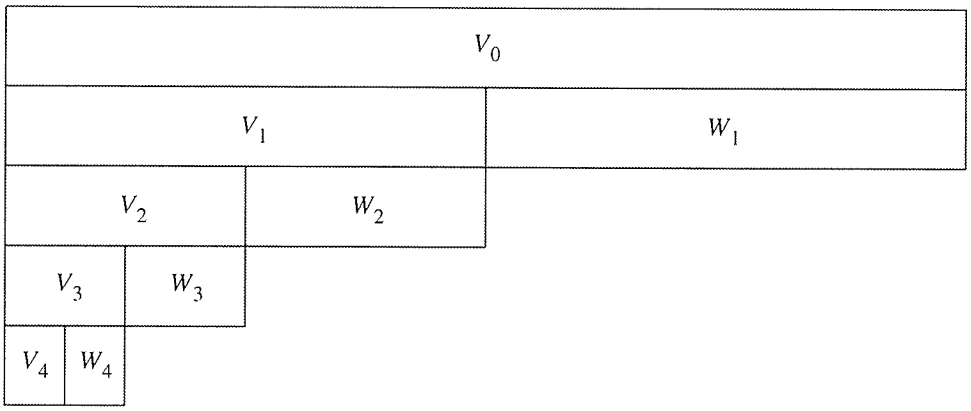


Fig. 3.5. MRA approximation and detail spaces.

3.4.5.2 MRA Analysis Algorithm

If we denote by A_j and D_j the approximation and the detail of the signal $f[x]$ at the resolution j then

$$A_j = \sum_{n \in \mathbf{Z}} \langle f, \phi_{j,n} \rangle \phi_{j,n} \tag{3.33}$$

$$D_j = \sum_{n \in \mathbf{Z}} \langle f, \psi_{j,n} \rangle \psi_{j,n} \tag{3.34}$$

and from Eq. 3.28

$$A_{j-1} = A_j + D_j . \tag{3.35}$$

where $a_n^j = \langle A_j, \phi_{j,n} \rangle$ and $d_n^j = \langle f, \psi_{j,n} \rangle$ are the approximation and detail coefficients respectively.

From the fact that $\phi(x) \in V_0 \subset V_{-1}$ and from Eq. 3.26, by the definition of $Span$ there exists $h[n]$, $n \in \mathbf{Z}$ such that

$$\phi(x) = \sum_{n \in \mathbf{Z}} h[n]\phi_{-1,n}(x) = \sqrt{2} \sum_{n \in \mathbf{Z}} h[n]\phi(2x-n) \tag{3.36}$$

Also, as $\psi(x) \in W_0 \subset V_{-1}$ and from Eq. 3.28, by the definition of $Span$ there exists $g[n]$, $n \in \mathbf{Z}$ such that

$$\psi(x) = \sum_{n \in \mathbf{Z}} g[n]\psi_{-1,n}(x) = \sqrt{2} \sum_{n \in \mathbf{Z}} g[n]\psi(2x-n) \tag{3.37}$$

Using the definition of a_n^j from Eq. 3.33, and combining with Eq. 3.37, the approximation coefficients can be represented as

$$\begin{aligned} a^j{}_n &= \langle f[x], \sum_{k \in \mathbf{Z}} h[k]\phi_{-1,n}(x)\rangle \\ &= \sum_{k \in \mathbf{Z}} h[k]\langle f[x], \phi_{j-1,k+2n}(x)\rangle \end{aligned} \tag{3.38}$$

by substituting $l = k + 2n$, Eq. 3.38 can be expressed as

$$a^j{}_n = \sum_{k \in \mathbf{Z}} h[l-2n]\langle f[x], \phi_{j-1,l}(x)\rangle \tag{3.39}$$

but $\langle f[x], \phi_{j-1,l}(x)\rangle$ is a_n^{j-1} therefore

$$a^j{}_n = \sum_{k \in Z} \tilde{h}[2n-k]a_n^{j-1} \tag{3.40}$$

where $\tilde{h}[n] = h[-n]$. Using the same arguments, the detail coefficients can be obtained by

$$d^j{}_n = \sum_{k \in Z} \tilde{g}[2n-k]a_n^{j-1} \tag{3.41}$$

where $\tilde{g}[n] = g[-n]$.

The MRA analysis algorithm is used to decompose the approximation and detail coefficients at resolution j from the approximation coefficients at resolution $j-1$. These equations are simply a convolution of the signal with the filter coefficients, followed by a downsampling by two. This recursive procedure is shown graphically in Fig. 3.6.

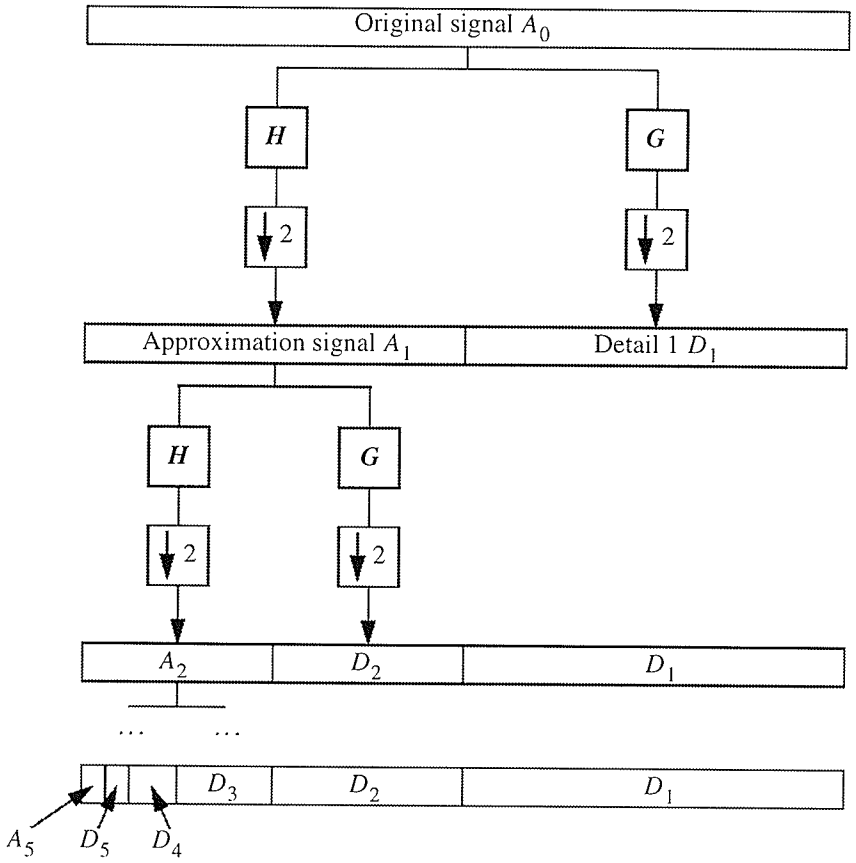


Fig. 3.6. Discrete wavelet transform using filter banks (From Dans01].

3.4.5.3 MRA Synthesis Algorithm

Using a similar argument as the MRA analysis algorithm, the MRA synthesis algorithm is

$$a_n^{j-1} = \left(\sum_k h[2n-k] a_k^j \right) + D^j \tag{3.42}$$

$$d_n^{j-1} = \sum_k g[2n-k] d_k^j \tag{3.43}$$

3.5 Summary

Data compression is the task of placing information in less space than originally allotted. Two popular compression techniques discussed in this chapter include scalar quantization, where only one symbol is considered at a time, and vector quantization where strings of symbols are considered. Since both of these techniques result in a loss of information, they are called lossy compression techniques. Another compression technique discussed was the quantization of transform coefficients. By transforming a signal into a transform domain, such as the Fourier domain or wavelet domain, the intersymbol correlation may result in energy compaction of the coefficients. If this were to occur, then a quantization of only the important coefficients could be performed to result in a higher quality compression technique.

The wavelet based transform technique will be used in further chapters to develop a raw SAR data compression technique. The wavelet transform decomposes a signal using scaled and shifted versions of a wavelet basis function. Stéphane Mallat developed a multiresolution analysis technique for the discrete wavelet transform using filter banks. This filter bank technique allows the fast computation of wavelet coefficients and will be used in the development of a VHDL wavelet transformer examined in future chapters.

CHAPTER 4

CURRENT RAW SAR DATA COMPRESSION TECHNIQUES

This chapter presents an overview of all known current raw SAR data compression techniques, organized into three sections: techniques using scalar quantization, techniques using vector quantization, and techniques in the transform domain. Several articles have been written comparing different raw SAR data techniques, interested readers are directed to [KuDC94], [SBBE94], [BeSM95], [PaCl99], and [ElBK01].

For raw SAR data aboard a spaceborne synthetic aperture radar with a data rate of R_{raw} to be transmitted within a bandlimited channel of bandwidth B_c, a compression ratio of at least B_{raw}/B_c:1 is necessary. For example, in the case of the Radarsat-1 satellite with a maximum B_{raw} of about 310 Mbits/s and a B_c of 105 Mbits/s, a compression ratio of about 3:1 is necessary.

Typically a downlink data rate of B_{raw} cannot be achieved, since it would require a large downlink transmitter and antenna subsystem that cannot be accommodated within the platform resources, given the large mass and power requirements of the SAR. The alternative is to reduce the system performance by modifying the system design or adding compression to the data collection procedure [CuMc91].

A compression technique to be employed in the compression of raw SAR data has two main objectives, (i) allow transmission of the raw SAR data in the bandlimited channel, and (ii) preserve as much of the information as possible. These two objectives are often competing, and finding a balance is only made more difficult due to the noise-like characteristics of the raw SAR data and the high entropy conservation.

4.1 Techniques Using Scalar Quantization

4.1.1 Quantization of the SAR Signal

As early as 1977, compression of raw SAR data has been the topic of research [LiBu77] due the enormous amount of data generated from a SAR system. Perhaps the simplest compression technique for raw SAR data is the quantization of data itself [LiBu77]. The technique discussed in [LiBu77] addresses the problem of storing and transmitting the raw SAR signal, before being processed into a SAR image, by digitizing the analog received SAR signal into digital raw SAR data. With the advancements in digital computing, the quantization of the raw SAR data aboard the radar itself using a A/D converter is now used on almost all SAR satellites.

4.1.2 Compression Using Precision Loss and Radiometric Loss

Perhaps the most significant constraint in the design of a SAR system is the bandlimited downlink channel. The bandwidth necessary for a given SAR system is

determined by three key factors, first, the number of bits used in the A/D conversion, second, the baseband bandwidth of the received SAR signal, and third, the PRF. The bandwidth of the received SAR signal, and the PRF are both determined by the geometry of the SAR radar. There are several options available to attempt a reduction in the required bandwidth by altering the radar geometry. Some of the available options are [CuMc91]:

(1) Increase the length of the SAR antenna and reduce the PRF at the expense of increased mass and degraded azimuth resolution;

(2) Reduce the bandwidth of the transmitted radar pulse and/or the oversampling factor to reduce the A/D converter sampling frequency at the expense of degraded range resolution;

(3) Reduce the swath width by increasing the incidence angle or changing the radar geometry at the expense of reduced ground coverage and increased geometric distortion from foreshortening and layover effects [CuMc91 Ch.8];

(4) Reduce the quantization performed by the A/D converter to fewer bits per sample at the expense of increased distortion noise and, therefore, degraded radiometric accuracy.

Of all the options mentioned above, Option 4 has been most typically used aboard SAR systems as the range and azimuth resolution are still maintained and it is

implemented by simply using an A/D converter with a larger quantization step size. The resulting radiometric loss produces images with less dynamic range and more overall noise. This technique is currently used aboard the ERS-1 [Atte91] and the Radarsat-1 satellites.

4.1.3 Lossless Coding Techniques

Due to the extremely high entropy of the raw SAR data, lossless compression techniques are usually employed. However, there has been some research done by [BrEK02] and [Bolle97], [Bolle98]. In [BrEK02], bit-planes of the raw SAR data and coded raw SAR data were investigated in an attempt to reduce the entropy of the data. It was found in [BrEK02] that none of the applied simple coding transforms yielded a significant entropy reduction, and it was suggested that a more advanced transform such as wavelets may produce better results.

The work of Bolle [Bolle97], [Bolle98] took a different approach by examining a reversible transform for the raw SAR data before entropy coding the coefficients. Bolle proposed the very interesting idea of applying a reversible transform to remove the chirp from the received SAR signal before transforming the data using the *discrete cosine transform* (DCT) and coding the coefficients, as shown in Fig. 4.1. This encoding scheme results in perfect reconstruction of the original SAR data with a compression ratio of 2.5-3.5 depending on the scene being imaged [Bolle98].

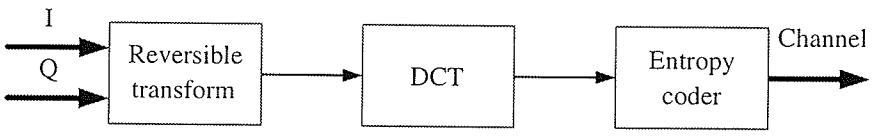


Fig. 4.1. Block diagram of Bolle encoder scheme.

4.1.4 Block Adaptive Quantization

Perhaps the most widely recognized form of raw SAR data compression is the *block adaptive quantization* (BAQ) developed by Kwok and Johnson of the NASA JPL [KwJo89]. As indicated by its name, the BAQ algorithm takes advantage of the slowly varying echo power in range from pulse-to-pulse to allow the block adaptation of the quantizer. This is accomplished by the determination of the statistics for each block of data samples, which in turn is used to adjust the quantizer levels. Since the statistical distribution of the real (I) and imaginary (Q) data samples is assumed to be zero-mean Gaussian with identical variance, the standard deviation is the only parameter necessary to describe the distribution.

The choice of blocksize is an important parameter in the BAQ algorithm. It is essentially a trade off between the number of samples necessary to ensure the block will have a Gaussian distribution, and an attempt to minimize the block size to obtain a stationary signal.

At its heart the BAQ algorithm is simply a non-uniform quantizer optimized for a Gaussian probability distribution where the threshold values are adapted on a block by block basis, and are derived from the variance of the block. A block diagram of the algorithm is shown in Fig. 4.2.

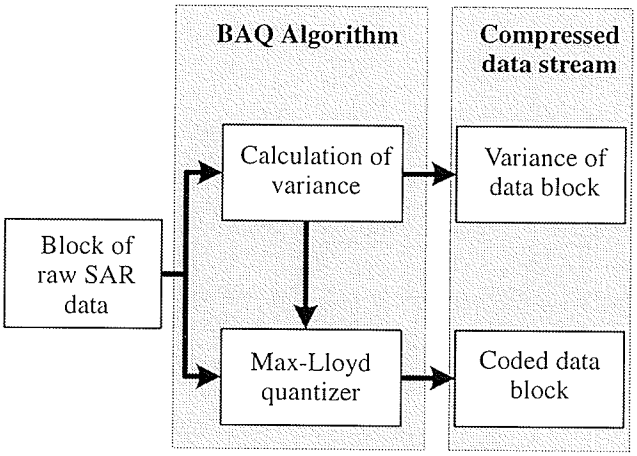


Fig. 4.2. Block diagram of the BAQ algorithm.

BAQ algorithm can be broken down into the following 4 steps as shown in Table 4.1.

Table 4.1 BAQ algorithm.

Step	Description
1	Acquire a block of raw SAR data.
2	Estimate the variance in each block.
3	Use the variance of the block to determine the optimum thresholds for a Max-Lloyd quantizer.
4	Compress the block of data using a Max-Lloyd quantizer.

Shown in Table 4.2 for several bit levels and compared to the Shannon bound, the BAQ algorithm is capable of obtaining the maximum achievable SQNR for a non-

uniform Gaussian optimized quantizer [Sayo96]. Although BAQ does produce acceptable results, it is still quite far from the Shannon bound and is, therefore, a sub-optimal raw SAR data compression technique.

4.1.4.1 Particularities of the BAQ Algorithm Used in the Magellan Mission

For the Magellan mission to Venus, the BAQ algorithm was modified slightly to reduce hardware complexity. One such modification was the use of the average signal magnitude instead of the variance, to generate the thresholds for the Max-Lloyd quantizer. This was accomplished by the equation

$$|\bar{I}| = |\bar{Q}| = 127.5 - \sum_{n=0}^{127} erf\left(\frac{n+1}{\sqrt{2\sigma}}\right) \tag{4.1}$$

which demonstrates a mapping of the variance of the block to the average magnitude. As shown in Fig. 4.3, the average signal magnitude is a close approximation to the standard deviation. A lookup table of values can then be used to calculate the optimum quantizer thresholds from this statistic. The performance difference between using the average magnitude statistic and the variance is small; however, the hardware complexity savings were significant [KwJo89].

The Magellan implementation of BAQ also used the statistic of the previous block to calculate the thresholds for the quantizer. This was done to minimize the amount of

buffering needed aboard the satellite, and was possible, as the raw SAR data has a slowly changing variance in both azimuth and range directions [KwJo89].

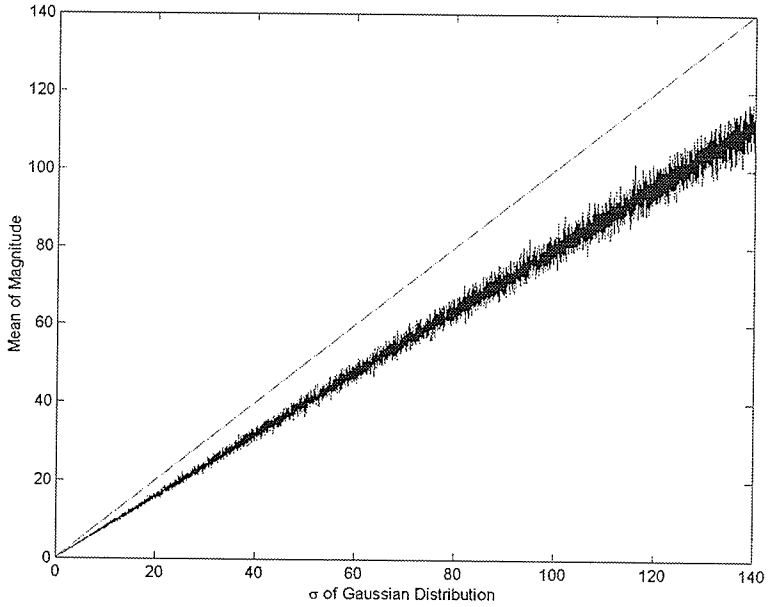


Fig. 4.3. Average magnitude versus standard deviation of a Gaussian signal.

4.1.4.2 Radiometric Preservation of BAQ

In the quantization of a Gaussian source using a Max-LLoyd quantizer, there is a loss of energy, and this is also the case for BAQ. In an attempt to preserve the radiometric energy of the reconstructed signal, the reconstruction levels of the Max-Lloyd quantizer are adjusted [KwJo89].

The average energy of a signal is given by

$$E[X^2] = \sigma^2 \tag{4.2}$$

but the average energy after quantization is given by

$$E[(q(X))^2] = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} y_i^2 f_x(x)dx \tag{4.3}$$

where $b_0 = -\infty$, $b_M = \infty$, and $f_x(x)dx$ is the PDF of source. The gain or loss of energy from the quantization G is

$$G = \frac{E[(q(X))^2]}{E[X^2]} = \sum_{i=1}^{M} \frac{y_i}{\sigma^2} \int_{b_{i-1}}^{b_i} y_i^2 f_x(x)dx \tag{4.4}$$

In the case of a 4-level symmetric quantizer, Eq. 4.4 can be simplified to

$$G = 2\left[\left(\frac{y_i}{\sigma}\right)^2 \int_0^{b_2} f(x)dx + \left(\frac{y_2}{\sigma}\right)^2 \int_{b_2}^{\infty} f(x)dx\right] \tag{4.5}$$

which can be further simplified for the case of a Gaussian signal to

$$G = 2\left[\frac{1}{2}\left(\frac{y_1}{\sigma}\right)^2 erf\left(\frac{b_2}{\sigma}\right) + \frac{1}{2}\left(\frac{y_2}{\sigma}\right)^2 \left[1 - erf\left(\frac{b_2}{\sigma}\right)\right]\right] \tag{4.6}$$

where the error function is $erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$, and y_1 and y_2 are the reconstruction values of the quantizer.

Using the Max-Lloyd reconstruction levels for 2 bits [Sayo96] into Eq. 4.6, we obtain $G \approx 0.8825$. Therefore, the average energy of the quantized raw SAR data is $0.8825\sigma^2$. In order to maintain the radiometric energy of the reconstructed raw SAR signal, the reconstruction levels of the Max-Lloyd quantizer would have to be raised by a factor of $1/0.8825$, resulting in the reconstruction levels of $\pm 0.52\sigma$ and $\pm 1.72\sigma$. This change in reconstruction levels also changes the SNR of the reconstructed signal for a 2-bit BAQ to 8.76dB as shown in Table 4.2.

Table 4.2 SQNR of BAQ [Sayo96], radiometric corrected BAQ [KwJo89], and the Shannon bound.

bits/sample	SQNR in dB		
	BAQ	BAQ with radiometric correction	Shannon Bound
2	9.30	8.76	12.04
3	14.61	14.46	18.06
4	20.23	20.18	24.08

4.1.5 Flexible BAQ

A slight variation on the BAQ algorithm is the *flexible BAQ* (FBAQ) [McCu95], [KuDC94]. The FBAQ algorithm works in the same manner as the Magellan BAQ except that the number of quantization bits for a given imaging pass is programmable from the ground station. FBAQ also only operates on a 1-D vector of samples, whereas the Magellan BAQ operates on a 2-D block of samples [KuDC94]. The finding of [KuDC94] showed no significant improvement in the performance of the

algorithm using a 2-D block, and therefore decided on the simpler 1-D block. The findings also show the performance of FBAQ to be the same as the Magellan BAQ. A block diagram of FBAQ is shown in Fig. 4.4.

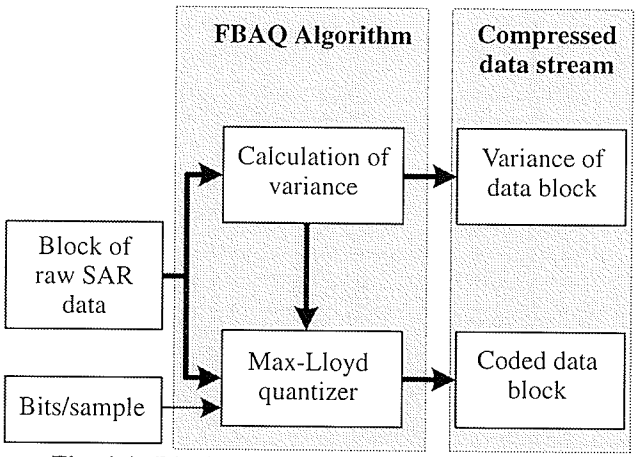


Fig. 4.4. Block diagram of the FBAQ algorithm.

4.1.6 Fuzzy Block Adaptive Quantization

The BAQ algorithm assumes that all blocks are Gaussian zero mean distributed, a small block, however, does not guarantee this supposition. The *fuzzy block adaptive quantization* (fuzzy-BAQ) attempts to improve this by the determination of the degree of membership of each block of data to a distinct Gaussian distribution [Benz94], [BeSM95], [SBBE94]. The membership will then determine which Gaussian distribution, and thereby which variance, best represents the data to give the lowest distortion. A block diagram of the fuzzy-BAQ algorithm is shown in Fig. 4.5.

In the fuzzy-BAQ algorithm, the estimation of the standard deviation and other quantization parameters is computed using a fuzzy system. The fuzzy input variables

are generated by the fuzzification of the crisp input variables, where the inputs are frequencies of the distinct values in the block. Using a rule-based algorithm, the fuzzy output variables are then calculated. Finally, the defuzzified output variables are used for the quantization.

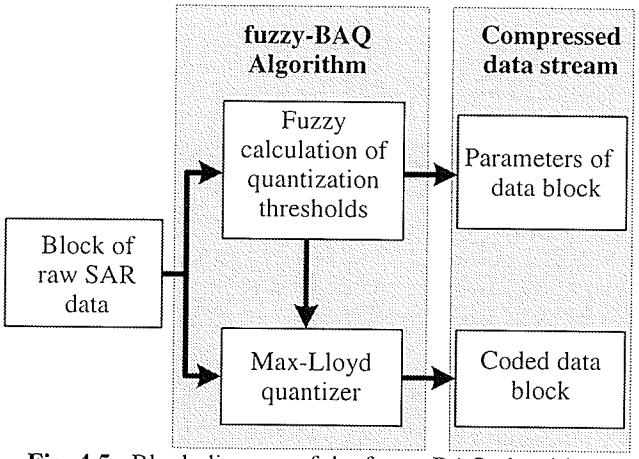


Fig. 4.5. Block diagram of the fuzzy-BAQ algorithm.

The fuzzy BAQ algorithm can be broken down into the following 4 steps as shown in Table 4.3.

Table 4.3 Fuzzy-BAQ algorithm.

Step	Description
1	Acquire a block of raw SAR data.
2	Use a fuzzy system to estimate the variance and other quantization parameters for the block of data.
3	Use these parameters in the to determine the optimum thresholds for the Max-Lloyd quantizer.
4	Compress the block of data using a Max-Lloyd quantizer.

Performance results from the fuzzy-BAQ algorithm developed in [BeSM95] are shown in Table 4.4 and are compared with BAQ.

Table 4.4 SQNR of fuzzy-BAQ [BeSM95] and BAQ.

bits/sample	SQNR in dB	
	fuzzy-BAQ	BAQ
2	10.7	9.30
3	n/a	14.61
4	n/a	20.23

4.1.7 Block Adaptive Histogram Equalization Quantization

The block adaptive histogram equalization quantization (BHEQ) [KuDC94], is an algorithm that first transforms a block of raw SAR data from a Gaussian distribution to a uniform distribution using the variance of the block. The transformation is achieved by computing the cumulative distribution function of the Gaussian distribution, and can be performed using look-up tables [KuDC94]. A block diagram of the BHEQ algorithm is shown in Fig. 4.6.

The BHEQ algorithm approaches the SQNR, without surpassing, of BAQ using a uniform quantizer. This lower SQNR is a result of the BHEQ minimizing the quantization error in the histogram-equalized domain, and not the quantization error in the original Gaussian distribution. Although BHEQ has a lower SQNR than BAQ, it has the benefit of a slightly less complex quantizer.

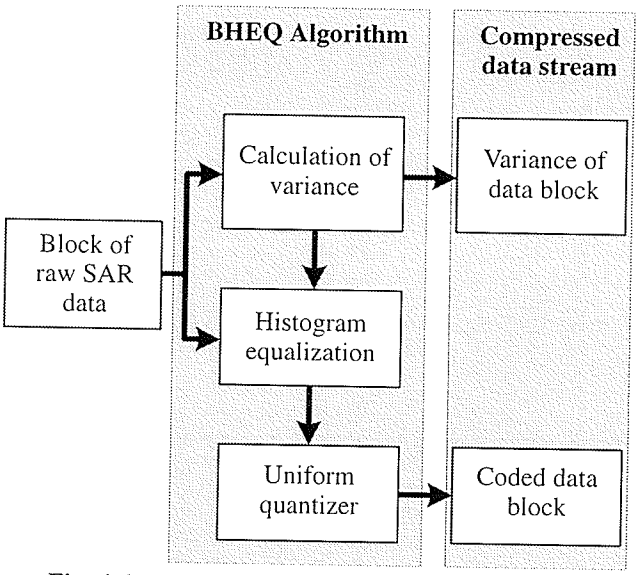


Fig. 4.6. Block diagram of the BHEQ algorithm.

The BHEQ algorithm can be broken down into the following 4 steps as shown in Table 4.5.

Table 4.5 BHEQ algorithm.

Step	Description
1	Acquire a block of raw SAR data.
2	Computation of the variance of the block of data.
3	Histogram equalization of the block of data.
4	Compression of the histogram equalized block of data using a uniform quantizer.

The SQNR results for BHEQ algorithm developed in [KuDC94] are shown in Table 4.6 and compared with BAQ. These SQNR results show that the BHEQ is consistently inferior to BAQ by, on average, 0.24 dB.

Table 4.6 SQNR of BHEQ [KuDC94], and BAQ.

bits/sample	SQNR in dB	
	BHEQ	BAQ
2	9.15	9.30
3	14.34	14.61
4	19.94	20.23

4.1.8 Block Adaptive Complex Quantization

In a typical SAR system, the raw SAR data is digitized as individual streams of real (I) and imaginary (Q) data samples. The *block adaptive complex quantization* (BACQ) functions by treating a pair of I and Q values as a complex sample, and then quantizing the sample using a quantizer with boundaries and reconstruction levels in 2-dimensional space [KuDC94]. Treating the data as a complex sample is essentially applying vector quantization to BAQ, where the codebook is comprised of vectors of length two. BACQ is discussed in this section as the complex value is treated as a single scalar.

To ensure that only one codebook is needed for the complex quantizer, each block of raw SAR data must have the same statistics. To accomplish this, the block can be histogram equalized [KuDC94], or each block can be simply divided by the variance of the block, essentially performing an 8-bit BAQ. A block diagram of the BACQ algorithm is shown in Fig. 4.7.

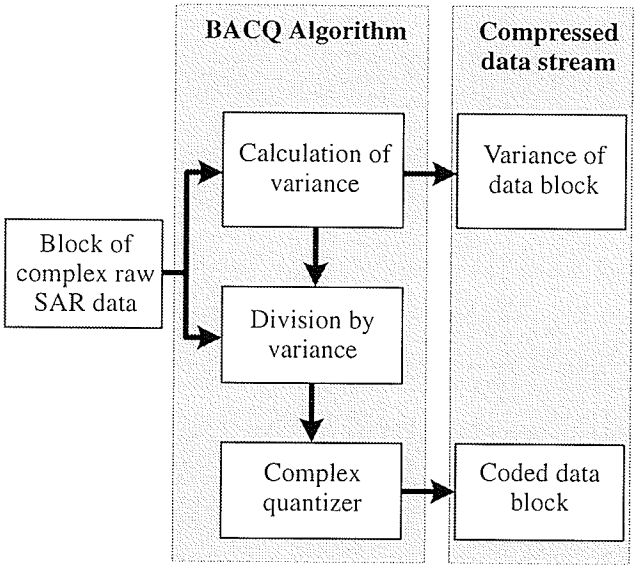


Fig. 4.7. Block diagram of the BACQ algorithm.

The basic BACQ algorithm can be broken down into the following three steps as shown in Table 4.7.

Table 4.7 BACQ algorithm.

Step	Description
1	Acquire a block of raw SAR data.
2	Remove the block to block variance using BAQ-8 or histogram equalization.
3	Determine the closest codevector to the 2-D sample.

The SQNR results for BACQ algorithm developed in [KuDC94] are shown in Table 4.8 and compared with BAQ. The only results presented in [KuDC94] for BACQ is for 2 bits/sample, and this result in inferior to BAQ by 0.15 dB.

Table 4.8 SQNR of BACQ [KuDC94], and BAQ.

bits/sample	SQNR in dB	
	BACQ	BAQ
2	9.15	9.30
3	n/a	14.61
4	n/a	20.23

4.1.9 Wide Bandwidth Block Adaptive Quantization

A relatively new variant of the BAQ algorithm is the *wide bandwidth BAQ* (WBBAQ), as proposed by Parkes [Park99], to address the bandwidth limitation of the BAQ algorithm. Parkes argues that although advances in IC technology will improve speed, the demands of future instruments will not be met and new BAQ architectures must be considered to provide the necessary additional performance. To address the need for increased bandwidth of a single BAQ IC, wide bandwidth BAQ examines the performance improvement by applying parallelism at the function, sample, and bit-level. WBBAQ does not improve the SQNR of BAQ, but suggests architectural improvements to the basic BAQ algorithm to increase the maximum data throughput [Park99].

4.1.10 Entropy-Constrained Block Adaptive Quantization

More recently an *entropy-constrained BAQ* (ECBAQ) has been proposed by [Algr00]. This technique promises to outperform BAQ in SQNR while still

maintaining a low level of computational complexity. The implementation of ECBAQ by [Algr00] also possesses the ability to control the output data rate by adjusting the level of quantization. This adjustment can be done without the need for developing new hardware, and can use non-integer rates.

The ECBAQ algorithm is simply an optimum quantizer, followed by an entropy coder. The quantizer in ECBAQ is an adaptive quantizer which adapts the thresholds based in the standard deviation of each block of data. This, in turn, also adapts the entropy coder. Unlike BAQ, where the optimum quantizer is non-uniform, in ECBAQ a uniform quantizer is optimum. In [Algr00] the entropy coder used is a Huffman coder [Sayo96], however, an adaptive arithmetic coder could also be used. A block diagram of the ECBAQ algorithm is shown in Fig. 4.8.

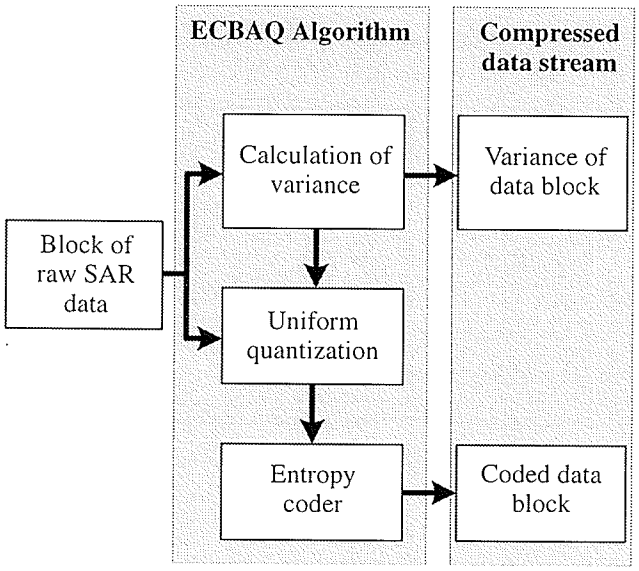


Fig. 4.8. Block diagram of the ECBAQ algorithm.

The ECBAQ algorithm can be broken down into the following 4 steps as shown in Table 4.9.

Table 4.9 BACQ algorithm.

Step	Description
1	Acquire a block of raw SAR data.
2	Calculate the variance of the block.
3	Uniformly quantize the samples using a uniform quantizer adapted to the variance of the block.
4	Entropy code the quantizer output to the desired rate.

The SQNR results for ECBAQ algorithm developed in [Algr00] are shown in Table 4.10 and are compared with BAQ. As can be seen from the SQNR results presented in Table 4.10, ECBAQ outperforms BAQ at all bit rates on average by 1.31 dB.

Table 4.10 SQNR of ECBAQ [Algr00] and BAQ.

bits/sample	SQNR in dB	
	ECBAQ	BAQ
2	9.67	9.30
3	16.17	14.61
4	22.23	20.23

4.1.11 Block Floating Point Quantization

The *block floating point quantization* (BFPQ) is principally similar to the BAQ algorithm, except that BFPQ uses a uniform quantizer, where as BAQ uses a non-

uniform quantizer [Hune89], [Hune90], [JoHW91]. The uniform quantization in BFPQ is simply a division by the quantization step adapted to each block, followed by a rounding. To speed up the calculation time, the quantization step can be rounded to the nearest power of 2, making the division a simple right shift.

The BFPQ algorithm derives its name from both the block of raw SAR data which are uniformly quantized, and, the subsets of available bits being selected by a predetermined algorithm, the equivalent of moving the "floating point" marker of the binary data. A block diagram of the BFPQ algorithm is shown in Fig. 4.9.

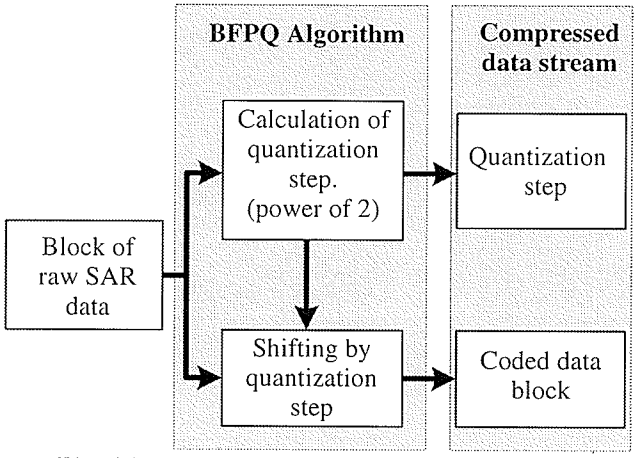


Fig. 4.9. Block diagram of the BFPQ algorithm.

The BFPQ algorithm can be broken down into 4 steps as shown in Table 4.11.

Table 4.11 BFPQ algorithm.

Step	Description
1	Acquire a block of raw SAR data.
2	Calculate the variance of the block.
3	Calculate of the quantization step rounded to the nearest power of 2.
4	Compress of the raw SAR data samples by simple binary shifting.

The BFPQ is particularly attractive because of its low computation complexity, which is even lower than that of BAQ. Like the BAQ algorithm, the BFPQ can adapt to the variation in power, however, the performance of BFPQ will always be inferior to that of BAQ because BFPQ uses a non optimal quantizer. The BFPQ has been used on several SAR systems, most notably the SIR missions [Hune89].

4.1.12 Block Adaptive Bit-Rate Control

The BAQ algorithm was developed to compress zero-mean, Gaussian distributed random data, where the dynamic range of the data changes slowly. In each block the same number of quantization levels are used to compress the data; therefore, in blocks with a high dynamic range, the quantization error is greater than in block with a low dynamic range. The method of *block adaptive bit-rate control* (BABC) was developed in order to shift parts of the code budget from blocks with a low dynamic range, to blocks with a high dynamic range under the constraint of an overall fixed code length for the complete data [ScSG01].

The BABC algorithm operates on blocks in the same manner as BAQ, applying a Max-Lloyd quantizer with the thresholds adapted to the variance of the particular block. However, BABC also adapts the number of quantization levels on a block-by-block basis. In many practical cases of SAR data acquisition, the dynamics of the data in the spectral domain are governed by the azimuth antenna pattern of the radar [ScSG01]. This leads to the signal dynamics of the SAR signal, and subsequently the raw SAR data, being approximately known in advance, making it possible to predetermine the bit-rates for each block. On-board the satellite these predetermined bit-rates can be applied to the blocks without considering the actual signal dynamics [ScSG01]. The BABC algorithm can be applied to any signal which has the same statistics as the SAR data, and can, therefore, be used on transformed raw SAR data as long as the data statistics remain the same.

It was found in [ScSG01] that the BAQ algorithm outperformed BABC in image domain SQNR, but it was suggested that this was due to the limited dynamic range of the test data. When BABC was preceded by a 2-D *fast Fourier transform* (FFT), yielding the FFT-BABC algorithm, the numerical results outperformed the BAQ algorithm significantly. It was also found that an appreciable performance increase could be obtained if the FFT-BABC was followed by an additional Huffman coder. A block diagram of the FFT-BABC algorithm is shown in Fig. 4.10.

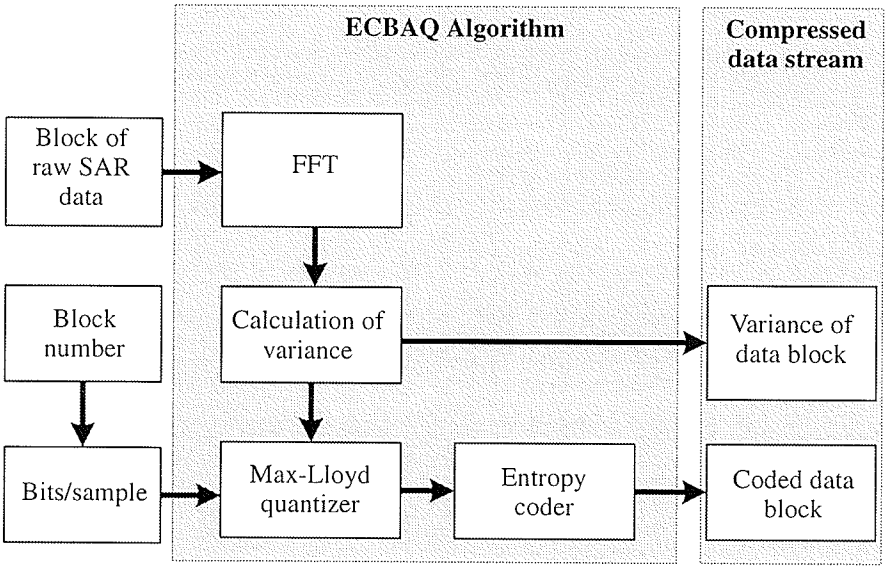


Fig. 4.10. Block diagram of the FFT-BABC algorithm.

The FFT-BABC algorithm can be broken down into the following 5 steps as shown in Table 4.12.

Table 4.12 FFT-BABC algorithm.

Step	Description
1	Acquisition of a 2D block of raw SAR data.
2	Application of a 2D FFT on the block of data.
3	Calculation of the variance of the block of data.
4	Compression of the block of data using a Max-Lloyd quantizer adapted to the variance of the block and using the number of quantization levels dictated by the predetermined bit-rate for the block.
5	Entropy code the quantized data using a Huffman coder.

4.1.13 Magnitude and Phase Block Adaptive Quantization

Raw SAR data is quantized in Cartesian complex form, but can be easily represented in polar form as well. *Magnitude and phase BAQ* (MPBAQ) applies an adaptive quantization technique to raw SAR data in polar form. Although the idea of MPBAQ has been alluded to in other work [Lebe95], [MGBB94], and developed mathematically by [Pear79], [PeSe79], and [Wils80], it has only been fully developed for SAR in [KuDC94].

When the complex SAR data is transformed to polar representation, the phase component can be modelled as uniformly distributed and the magnitude as Rayleigh distributed [Good75]. Separate quantizers are then required for each distribution to quantize the data optimally for a given number of thresholds. The optimum bit allocation for the magnitude and phase components has been developed by [Pear79] and [PeSe79] for a Gaussian source. A block diagram of the MPBAQ algorithm is shown in Fig. 4.11.

It has been suggested in [Lebe95] that MPBAQ would be too computationally complex to implement on-board a satellite, due to the Cartesian-to-polar transformation. Although this is true if done after the SAR signal is digitized, it is not the case if it is done using analog electronics on the received SAR signal. The signal could easily be transformed into magnitude and phase rather then inphase and quadrature prior to digitization by the A/D converter.

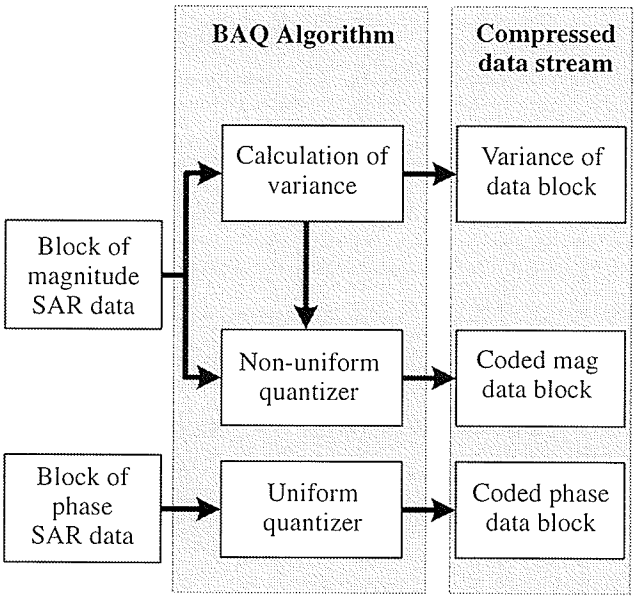


Fig. 4.11. Block diagram of the MPBAQ algorithm.

The MPBAQ algorithm can be broken down into the following four steps as shown in Table 4.13.

Table 4.13 MPBAC algorithm.

Step	Description
1	Acquisition of a block of raw SAR data in polar format.
2	Calculation of the variance for the magnitude data blocks.
3	Use the variance of the blocks to determine the optimum quantization levels for the magnitude quantizer.
4	Quantize the phase block using a uniform quantizer and the magnitude block using a non-uniform quantizer for a Rayleigh distribution.

Research by [KuDC94] varied the bit allocation to the magnitude and phase components as shown in Table 4.14 with the best SQNR performance for the given number of bits per sample shaded. Performance results of MPBAQ, using the bit

allocations of Table 4.14, is shown in Table 4.15 and is compared with BAQ. Table 4.15 shows that MPBAQ consistently performs worse than BAQ by 0.05 dB on average.

Table 4.14 SQNR in dB of MPBAQ for various bit allocations [KuDC94].

Number of bits/sample for encoding the magnitude	Number of bits/sample encoding phase				
	1	2	3	4	5
0				6.48	6.63
1			**9.19**	10.70	11.19
2		6.63	**11.41**	**14.57**	15.93
3	1.38	6.89	12.44	**17.21**	**20.22**
4	1.38	6.98	12.79	18.40	**23.11**
5	1.39	7.00	12.89	18.78	24.38

Table 4.15 SQNR of MPBAQ [KuDC94] and BAQ.

bits/sample	SQNR in dB	
	MPBAQ	BAQ
2	9.19	9.3
3	14.57	14.61
4	20.22	20.23

4.1.14 Predictive Coding

Due to the very low sample-to-sample correlation of raw SAR data, some have concluded that predictive coding is not a viable technique for raw SAR data compression [Lebe95]. Despite this, some work in this area has been done with

positive results [MaOl02]. It has been noted in [MaOl02], that raw SAR data is more predictable in range than in azimuth, leading [MaOl02] to develop a compression algorithm based on *differential pulse code modulation* (DPCM), which is able to capture the correlation of raw SAR data along range lines.

If during the compression, the energy of a block of raw SAR data to be predicted is the same, only one set of prediction coefficients is necessary. This is the method used by [MaOl02] in their *range DPCB BAQ* (RDPCM-BAQ). A block diagram of the RDPCM-BAQ algorithm is shown in Fig. 4.12.

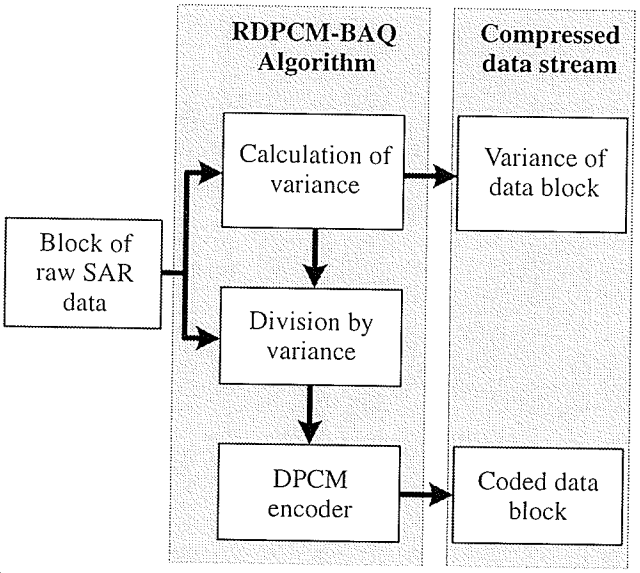


Fig. 4.12. Block diagram of the RDPCM-BAQ algorithm.

The RDPCM-BAQ algorithm can be broken down into the following four steps as shown in Table 4.16.

Table 4.16 RDPCM-BAQ algorithm.

Step	Description
1	Acquisition of a block of raw SAR data in range.
2	Calculation of the variance of the block.
3	Normalization of the block by division by the block's variance.
4	Compression of the block of data using DPCM.

From an implementation perspective, the RDPCM-BAQ is very attractive as it functions on lines of raw SAR data. Because of this, little-to-no memory would be required to buffer the data before compression could commence, as is necessary for techniques requiring a 2D block of data. Results for the RDPCM-BAQ technique using a predictor order of 15 are shown in Table 4.17 and compared with BAQ. These SQNR results show that RDPCM-BAQ outperforms BAQ by an average of 0.62 dB.

Table 4.17 SQNR of RDPCM-BAQ [MaOl02] and BAQ.

bits/sample	SQNR in dB	
	RDPCM-BAQ	BAQ
2	9.9	9.30
3	15.25	14.61

4.2 Techniques Using Vector Quantization

From communications theory, it can be shown that vector quantization will always perform at least as well as scalar quantization [Gray84]. It is then a natural progression for vector quantization to be applied in the compression of raw SAR data. There has

been much research that applies vector quantization to raw SAR data compression, most of which has approached the topic from two perspectives. The first group of researchers developed techniques to apply vector quantization to the entire data frame at once [Arno87], [JVFC88], [MGBB94], [RACC88], and the second group of researchers applied vector quantization on blocks of raw SAR data [Lebe95], [LMBL95], [BoBl93], [SBBE94], [BeSM95], [PaCl99].

4.2.1 Vector Quantization

The appeal of vector quantization in other applications has led researchers to apply it in the compression of raw SAR data. The techniques used in [Arno87], [JVFC88], [MGBB94], and [RACC88], are fundamentally the same, whereby a vector quantizer is used to compress an entire raw SAR data frame at once. The standard techniques of LGB and lattice vector quantization have been used to design the codebooks [Arno87], [MGBB94], in addition to determining the optimal codebook size [JVFC88]. Although no raw SAR data SQNR results are described in these papers, it is noted in [MGBB94] that vector quantization outperformed BAQ, especially at low bit rates. It was also noted in [MGBB94] that the lattice vector codebook design was superior to the LBG codebook.

4.2.2 Block Adaptive Vector Quantization

It has been shown that raw SAR data has a slow changing power in both range and azimuth directions [KwJo89]. By applying vector quantization on a block-by-block basis, the vector quantizer codebook can be more specific to the statistics of the given block. If the block of raw SAR data is normalized to have the same variance, then only one codebook is needed for all blocks. This sequence is basically a BAQ with the output followed by a vector quantizer. The resulting algorithms, which are all fundamentally the same, are the *block gain adaptive vector quantization* (BGAVQ) [Lebe95], [LMBL95], the *block adaptive vector quantization* (BAVQ) [BoBl93], [SBBE94], [BeSM95], and the *gain-shape vector quantization* (GSVQ) [PaCl99]. A block diagram of the BAVQ is shown in Fig. 4.14. Performance results for the BGAVQ algorithm from [Lebe95] for a vector of length 4 and compared with BAQ, are shown in Table 4.19. It has been noted by most relevant authors that the computation complexity increase for a block vector quantization technique is slightly larger then BAQ but the performance results justify this increase.

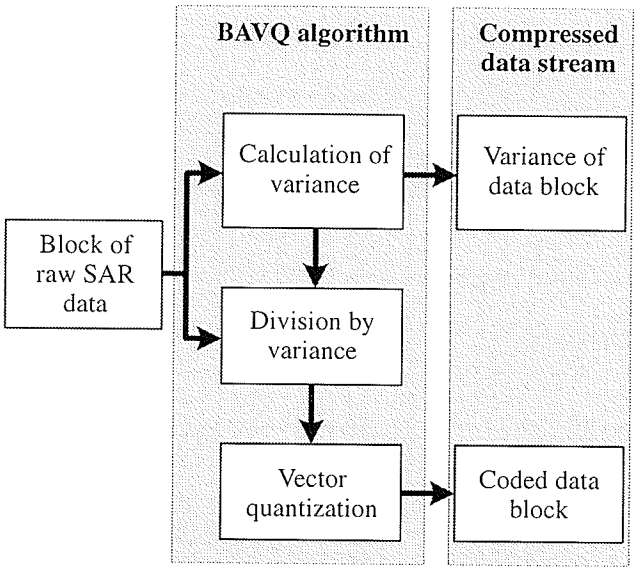


Fig. 4.13. Block diagram of the BAVQ algorithm.

The BAVQ algorithm can be broken down into the following four steps as shown in Table 4.18.

Table 4.18 BAVQ algorithm.

Step	Description
1	Acquisition of a block of raw SAR data in range.
2	Calculation of the variance of the block.
3	Normalization of the block by division by the block's variance.
4	Quantization of each vector of samples using a vector quantizer and a predesigned codebook.

4.2.2.1 Block Adaptive Vector Quantization using Magnitude and Phase Data

Just as scalar quantization can be applied to raw SAR data in both Cartesian and polar representations, so can vector quantization [Lebe95], [MoBl93]. The resulting

algorithm is the *block gain adaptive unrestricted polar quantizer* (BGAUPQ). Results of the magnitude and phase technique vary, as results from [Lebe95] show a lower SQNR where as [MoBl93] show a higher SQNR. Results of the BGAUPQ from [Lebe95] are shown in Table 4.19 and compared with BAQ. This table shows that both the BGAVQ and BGAUPQ outperform BAQ, with BGAVQ demonstrating the greatest SQNR results of the raw techniques surpassing BAQ by 0.33 dB at 2 bits/sample.

Table 4.19 SQNR of BGAVQ [Lebe95], BGAUPQ [Lebe95], and BAQ.

bits/sample	SQNR in dB		
	BAQ	BGAVQ	BGAUPQ
2	9.30	10.03	9.55

4.2.3 Trellis Coded Quantization

Similar to the idea of vector quantization is the *trellis coded quantization* (TCQ), which has been applied in the compression of raw SAR data by several researchers [Owen97], [OMHK97], [OMHK99]. One of the attractive features of TCQ is that it can process very long vectors with a computational complexity that is independent of vector length. This is in sharp contrast to full search vector quantization which requires exponential complexity for increasing vector length.

Two TCQ techniques have been applied in the compression of raw SAR data, the *trellis coded vector quantization* (TCVQ), and the *universal TCQ* (UTCQ). UTCQ is a

refinement over TCQ in that it does not require trained and stored codebooks, and allows noninteger encoding rates. However, UTCQ uses an arithmetic coder [Sayo96] and as such has variable length codes making it very sensitive to bit error rates. TCVQ addresses this problem by using fixed length codes. More detailed information on TCQ can be found in [FiMW91]. Performance results for TCVQ and UTCQ from [OMHK99] are shown in Table 4.20 and compared with BAQ. These results show that both TCVQ and UTCQ outperform BAQ at all bit rates with UTCQ yielding the greatest gains. The SQBR results show that UTCQ outperforms BAQ by 1.86 dB on average and increases in SQNR gains with increasing bit rates.

Table 4.20 SQNR of TCVQ [OMHK99], UTCQ [OMHK99], and BAQ.

bits/sample	SQNR in dB		
	BAQ	TCVQ	UTCQ
1	4.39	5.19	5.22
2	9.30	10.69	11.30
3	14.61	16.29	17.37

4.3 Techniques in the Transform Domain

4.3.1 Transform BAQ

The BAQ algorithm was developed to compress a zero mean Gaussian signal, such as raw SAR data. This means that BAQ can also be applied to transformed raw SAR data if it has the same statistics, as is the case of an orthogonal transform. This has led

to substantial research in the development of transformed based BAQ algorithms [SBBE94], [BeSM95], and [FiBM99]. The research attempted several transforms such as the *discrete cosine transform* (DCT), the *Walsh-Hadamard transform* (WHT), and the *fast Fourier transform* (FFT).

All of these transform based BAQ techniques work in the same manner as shown in the block diagram in Fig. 4.13. The techniques are simply a transform on a block of raw SAR data followed by a BAQ of the coefficients.

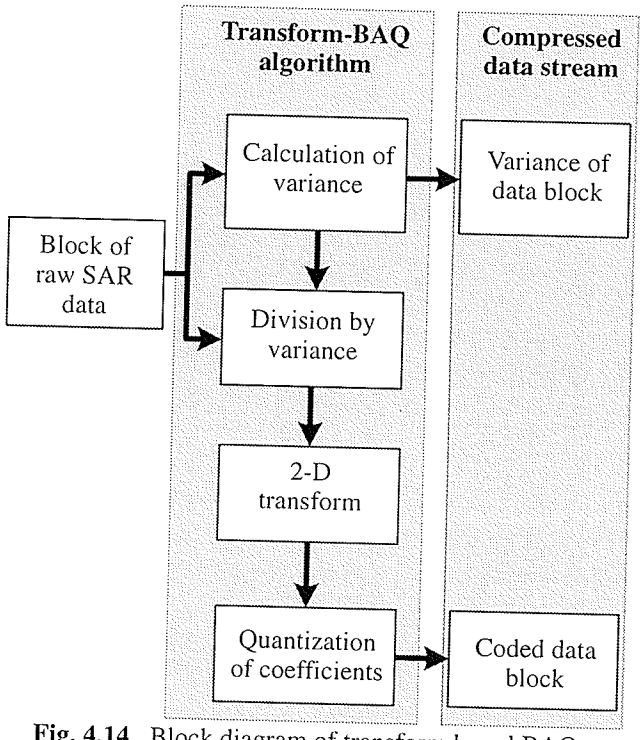


Fig. 4.14. Block diagram of transform based BAQ.

The transform based BAQ algorithms can be broken down into the following five steps as shown in Table 4.21.

Table 4.21 Transform-BAQ algorithm.

Step	Description
1	Acquisition of a 2-D block of raw SAR data.
2	Calculation of the variance of the block.
3	Normalization of the block by division by the block's variance.
4	2-D transformation of the block of data.
5	Quantization of the transform coefficients using BAQ.

Although no SQNR performance metrics in the signal domain are given in the papers, it has been noted in [BeSM95] that the FFT-BAQ algorithm had the best performance but the highest computational complexity.

4.3.1.1 Particularities of the FFT-BAQ Algorithm

The idea presented in [BeSM95] for FFT-BAQ is to adapt the data compression to the energy variation of the frequency envelope of the raw SAR data, thereby quantizing the regions with higher energy and with more bits. Additionally, the frequency region outside the processed bandwidth does not need to be coded at all. This then requires that the BAQ used on the transformed data has a variable compression ratio.

[BeSM95] claims that in practice, the processed bandwidth corresponds to approximately 85% of the full signal bandwidth, allowing an immediate data reduction of 15% with no image degradation. The bit allocation scheme for the BAQ used on the transformed data in the FFT-BAQ from [BeSM95] is shown in Fig. 4.15.

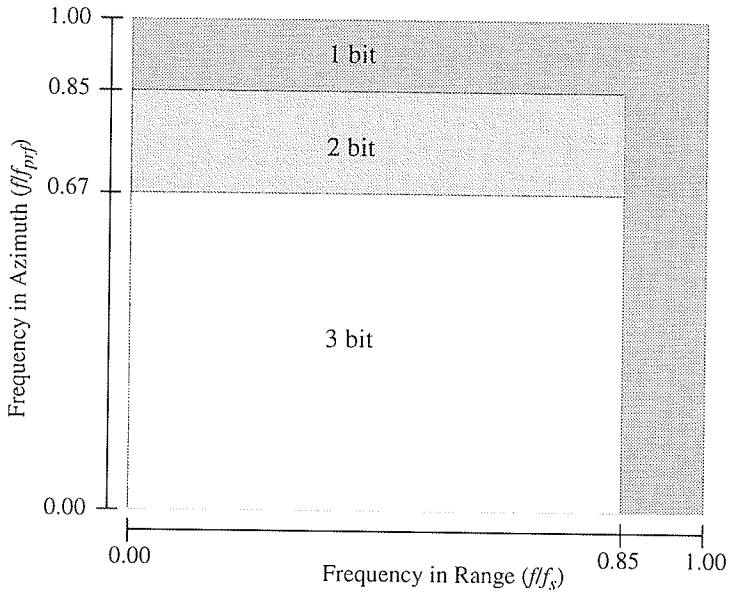


Fig. 4.15. Bit allocation scheme for FFT-BAQ. (From [BeSM95])

4.3.2 Compression Using Wavelets and Wavelet Packets

The growing popularity of wavelets, has led several researchers to take an interest in applying wavelets to the compression of raw SAR data. There have been two approaches to raw SAR data compression using wavelets, those compressing the entire data frame at once [PaSc99], [PaSB99], [PaSc00], and those applying the wavelet transform to blocks of data [ETHB02], [MaOP02].

Both approaches have advantages and disadvantages, but it is the wavelet-BAQ which has the highest probability of practical success as the entire data frame does not have to be stored in memory before the compression can begin. The wavelet compression of the entire data frame as done by [PaSc99] and [PaSB99] consists of applying a DWT to the data frame and thresholding the coefficients. The insignificant

coefficients are quantized to zero while the significant coefficients are uniformly quantized and transmitted along with the significance map noting the significant coefficients. The work done by [PaSc99] and [PaSB99] note that their compression yields better performance then BAQ, but only provide SQNR results in the image domain, and not the signal domain.

A slight variation of this idea uses wavelet packets rather then wavelets [PaSc00]. As previously discussed, the entire data frame is compressed at one time with the coefficients thresholded and then quantized. The data transmitted in this application is the significance map and the quantized coefficients. Results from [PaSc00] on their wavelet packet compression technique is shown in Table 4.6 and compared with the SQNR of BAQ. Although these results show a performance advantage over BAQ by an average of 1.43 dB, the computational and memory requirements for the wavelet packet technique on the full data frame make it impractical for on-board applications.

Table 4.22 SQNR of wavelet packets compression [PaSc00] and BAQ.

bits/sample	SQNR in dB	
	Wavelet packet	BAQ
1	5.44	4.39
2	11.11	9.30

4.3.2.1 Wavelet Block Adaptive Quantization

An alternative approach to applying wavelets directly to the entire data frame is to apply it to blocks of data, and adapting the quantization to the specifics of each block

thereby creating the wavelet-BAQ. The work by [ETHB02] and [MaOP02] is fundamentally the same, whereby blocks of raw SAR data are transformed using DWT and the coefficients quantized using a Max-Lloyd quantizer for a Gaussian distribution. The main difference is that the wavelet-BAQ by [MaOP02] first normalizes each block of data so that the variance of each block being transformed by the MRA is the same. Using this technique, the bit-allocation need only be determined once, as it is related to the variance of the original block. A block diagram of the [MaOP02] wavelet-BAQ is shown in Fig. 4.16.

Results from [ETHB02] are shown in Table 4.23 and compared with BAQ. The table shows the results for the wavelet-BAQ using the Haar, *Battle-Lemarié linear* (BLL), *Battle-Lemarié quadratic* (BLQ), and two Daubechies wavelets (D-4 and D-20). These results show no significant difference from basis to basis and no improvement over BAQ as their SQNR performance remains about the same. [MaOP02] also discuss this issue in their results noting that the wavelet-BAQ produced comparable results to BAQ at all bit levels.

Table 4.23 SQNR of BAQ and wavelet-BAQ [ETHB02].

bits/sample	SQNR in dB					
	BAQ	Wavelet-BAQ				
		Haar	BLL	BLQ	D-4	D-20
1	4.39	4.35	4.35	4.35	4.36	4.35
2	9.30	8.14	8.01	8.00	8.01	8.00
3	14.61	14.55	14.52	14.59	14.53	14.50

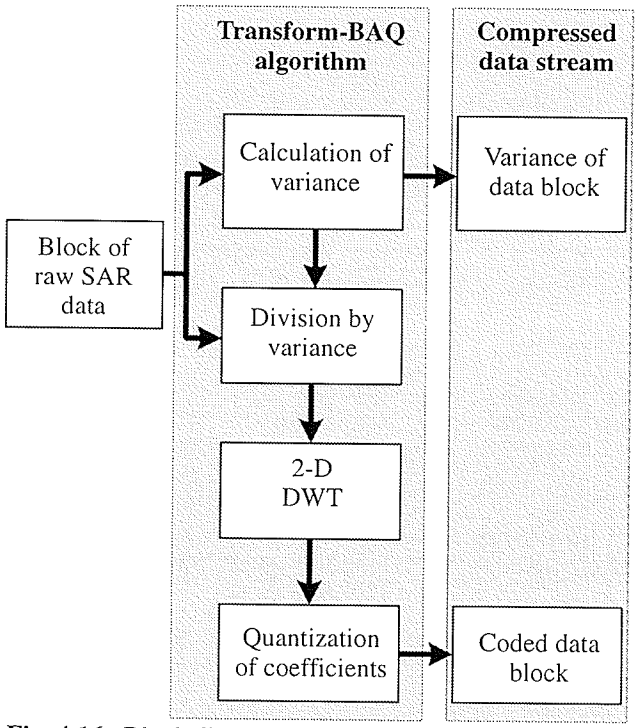


Fig. 4.16. Block diagram of the wavelet-BAQ algorithm.

The [MaOP02] wavelet-BAQ algorithm can be broken down into the following five steps as shown in Table 4.24.

Table 4.24 Transform-BAQ algorithm.

Step	Description
1	Acquisition of a 2-D block of raw SAR data.
2	Calculation of the variance of the block.
3	Normalization of the block by division by the block's variance.
4	Transformation of the block using a 2D MRA.
5	Quantization of the coefficients using a predetermined bit allocation and a Max-Lloyd quantizer for a Gaussian distribution.

4.4 Summary

Since at least 1977, researchers have been working on ways to compress raw SAR data, a task not easily accomplished. Throughout this research, many techniques have been developed and attempted, but the BAQ algorithm developed by NASA JPL remains the standard. Recent work in the vector quantization of raw SAR data and transform based technique has shown promising results, along with work in predictive coding. It is important to note however, that most techniques achieve SQNR performance results that are not significantly different than BAQ, and are still quite far from the theoretical Shannon bound.

Of the techniques developed and discussed in this chapter, transform based techniques appear to be the most promising as they are able to decorrelate the raw SAR data, removing any redundancy that may exist. One of the problems with the current transform based techniques is that the necessary computational complexity and memory requirements prohibit the implementation the technique on-board a satellite. This issue will be addressed in the following chapters by developing a low complexity raw SAR data compression technique based on wavelets.

CHAPTER 5

DESIGN OF A WAVELET-BAQ TECHNIQUE

The compression of raw SAR data has been the focus of research for many years, and many novel approaches have been developed. One of the difficulties with developing any new raw SAR data compression technique is the noise like characteristics of the data. This makes conventional image compression techniques ineffective, and produces poor quality results.

A potential technique that has shown exceptional results in other fields of signal and data compression is the wavelet transform. The wavelet transform has the ability to decompose a signal using a specific function, called the mother wavelet, as the basis. Using the MRA algorithm developed by Mallat, the wavelet transform can be efficiently performed using filter banks, thus resulting in an algorithm easily implemented on a computer.

This chapter describes a technique for raw SAR data compression using the wavelet transform. The chapter presents the criteria by which a new technique is based, and the implementation specifications for the new wavelet technique detailing how the MRA algorithm is applied to the compression of raw SAR data. Details discussed are the implementation of the discrete convolution, quantization of the MRA coefficients and the choice of wavelet basis. This chapter also presents the raw SAR data pre-conditioning technique used in this work to create the 8-bit raw SAR data sets

from the original 4 and 5-bit data sets. These test sets are necessary to simulate the raw SAR data compression technique with data of the same type it would see if it was placed about the radar satellite.

5.1 Criteria for a new Raw SAR Data Compression Technique

In the development of a new practical on-board raw SAR data compression scheme, several criteria should be examined. For this work the following criteria will be used for the development of a new raw SAR data compression technique.

(1) The new algorithm should have the potential to produce less distortion due to quantization than BAQ.

(2) The new technique must be of low computational complexity and in high throughput.

The first criterion is fairly self evident, no new compression scheme should aim to perform lower than the current standard compression scheme, namely BAQ. The second criterion is one that many researchers have ignored in their research, but if an algorithm is impossible to implement in practice it is not a candidate for on-board use. A prime example is the Karhunen-Loéve transform (KLT) which, in theory, results in the greatest decorrelation of raw SAR data, but since it requires a very large number of calculations per block it is unsuitable for application on-board a satellite.

The second criterion presents two main constraints about any potential implementation. Addressing these constraints amounts to reducing the buffering required by the implementation by processing the data directly from the ADC. This requires that the resulting algorithm operate on the data at the sampling rate of the A/D and operate on the data as it arrives.

The wavelet transform has previously shown promising results in many other fields of signal and data compression [Daub92], and it is the focus of this work to apply it to the compression of raw SAR data. The MRA algorithm has also been shown to be effectively implementable in pipelined hardware [Parh99]. A wavelet based technique, if producing results greater than BAQ, would satisfy all requirements formentioned as the technique is practically implementable using filter banks, previous hardware implementations have been shown to have high throughput [Parh99], and the implementation can be pipelined so that it is low on memory requirements and buffering.

The implementation of the MRA algorithm presented in Chapter 4 immediately poses the following questions: (i) how exactly is the MRA algorithm implemented, (ii) how should the MRA coefficients be quantized, and (iii) what wavelet basis should be used. These questions will be discussed and answered in this chapter using experimental results to provide the design specification for the hardware implementation in the next chapter.

5.2 Pre-conditioning of the Raw SAR Data Test Sets

In order to measure the performance of any new raw SAR data compression technique, some form of SAR data test set is necessary. From the literature, there have been three main types of test sets used, simulated data, real raw SAR data, and pre-conditioned raw SAR data. The researchers that use simulated data either produce random numbers from a Gaussian source, or use a SAR data simulator to produce the data. While the Gaussian source simulated data is designed to have the same statistics as the basic raw SAR data model from [Good75], it is not capable of revealing any underlying data that may exist in real data. The simulator produces data using a much more complex model than simply a Gaussian source, and has proven useful in other work [Lebe95]. Unfortunately programming a quality SAR simulator requires significant knowledge and time, and existing SAR simulators are not readily available to researchers not closely affiliated with a space agency.

The ideal situation is where 8-bit quantized raw SAR data could be used for testing. Using such data, the new compression technique could be applied to the very data that would appear after the digitization on-board the satellite. Unfortunately, such data is not readily available to most researchers as it is not a standard SAR data product available from SAR ground stations. Instead, 4 to 5 bit quantized data, or BAQ quantized data is readily available as this is the raw data received by the ground stations from the satellite. While this data is real raw SAR data, applying a

compression technique to this data will not give a true measure of the technique's performance since the compression technique is designed to compress the 8-bit data from the A/D converter.

The data used in this work is ERS-1, ERS-2, and Radarsat-1 data obtained from the Alaska SAR facility. The data compression technique used on-board these satellites is dynamic range reduction. This means that of the 8-bits available, if an 8-bit A/D converter had been used, only 5-bits, or 4-bits for Radarsat, was transmitted. The problem is then, how to take the 4 and 5-bit quantized raw SAR data and make it 8-bit. The resulting data, which is based on real raw SAR data, has 3 or 4 bits of additional simulated data added and is known as pre-conditioned raw SAR data [McCu95].

This problem of generating 8-bit test sets has been investigated by several researchers, however, no paper explicitly describes the pre-conditioning technique used. Instead, most refer to the proprietary ESA document developed by MacDonald Detwiler [Dutk93]. Unfortunately, due to a contractual agreement, which must be made to obtain the technical report, the exact details of their pre-conditioning technique can not be published.

5.2.1 Selection Criteria

When selecting a technique to use for the pre-conditioning of raw SAR data, the statistics of the pre-conditioning data should be equal, or related by a known scaling

factor, to the original raw SAR data. The statistics investigated in [Dutk93] are, (i) skewness, which is expected to be equal to the original data, (ii) kurtosis, which is expected to be equal or similar to the original data, (iii) standard deviation, which is expected to be equal or similar to the standard deviation or the original SAR data multiplied by a scaling factor, (iv) the entropy of the of pre-conditioned data should be lower than 8 by the same factor as the original data is from its quantized value, and (v) the frequency of saturations should be less in the pre-conditioned data than in the original data. An additional requirement that will be added for this work is that the shape of the histogram should be smooth. The tails of the histogram are due to the limited dynamic range of the A/D converter. Intuitively, this should cause only the maximum and minimum value to be elevated and this elevation should decrease as the number of bits, and hence the dynamic range, increases.

Analysis of the statistics can be used to select a pre-conditioning technique that produces data with properties equivalent to those which would have been obtained if an 8-bit A/D converter had been used. To illustrate the desired shape of the histogram, consider a unit variance Gaussian random process placed in 256 equal spaced bins between -3 and 3, as shown in Fig. 5.1. The tails shown in this figure represent the samples that occurred outside [-3,3], and corresponds to the same issue of the A/D converter on the satellite. Notice that this histogram is relatively smooth, corresponding to the expected shape and has tails only at the extremities.

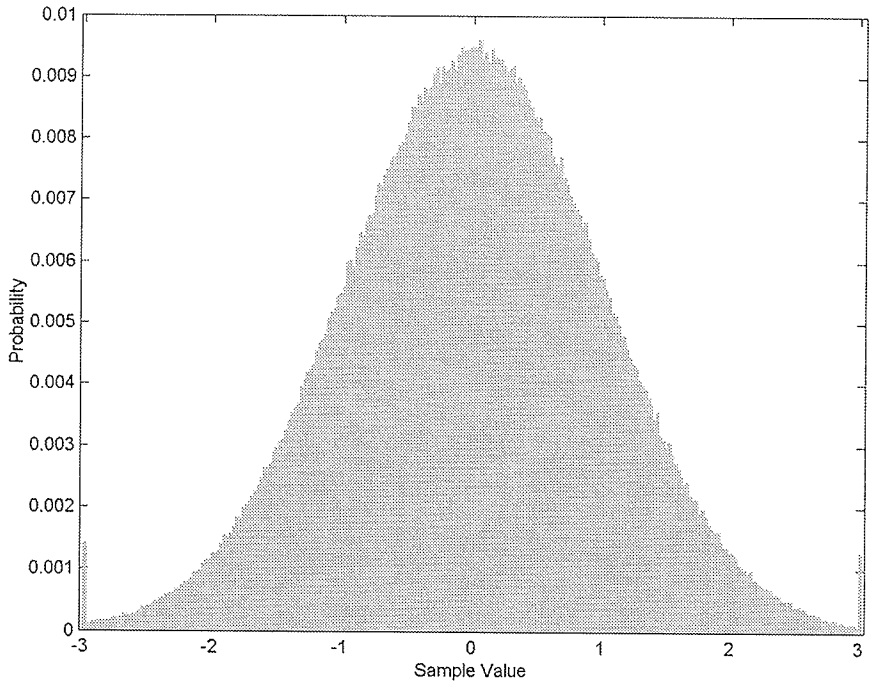


Fig. 5.1. Histogram of Gaussian source with 256 equal space bins between -3 and 3.

5.2.2 Pre-condition Techniques

The raw SAR data product is available from SAR ground stations by their industry name "level zero CEOS data". The data files contain the raw SAR data in altering samples of real (I) and imaginary (Q) stored in unsigned byte format and are assumed to have a mean 15.5 or 7.5, depending on the number of quantization bits. Once pre-conditioned, the data must be rewritten in this same format for later processing into an image by the SAR processor.

Since the problem of data pre-conditioning is the addition of the missing bits of information, standard noise distributions can be used to add the additional data. The basic procedure for additive noise data pre-conditioning is shown in Table 5.1.

Table 5.1 Basic additive noise pre-conditioning procedure.

Step	Description
1	Read a range line from the original raw SAR data test file and convert each sample read to a floating point number.
2	Subtract the mean from the data line.
3	Multiply each sample by α, where α is the difference of 8 and the number original quantization bits.
4	Add α bits of noise to each sample.
5	Convert the data from floating point to byte format.

Using the additive noise technique for data pre-conditioning with uniform and white noise sources produces data with histograms similar to those shown in Fig. 5.2 and Fig. 5.3. Without even calculating other statistics for these data sets it can be seen that they do not satisfy all the criteria. The data set created using additive white noise has increased the variance but the histogram is not smooth at all. In fact, the addition of white noise has simply produced several small Gaussian distributions.

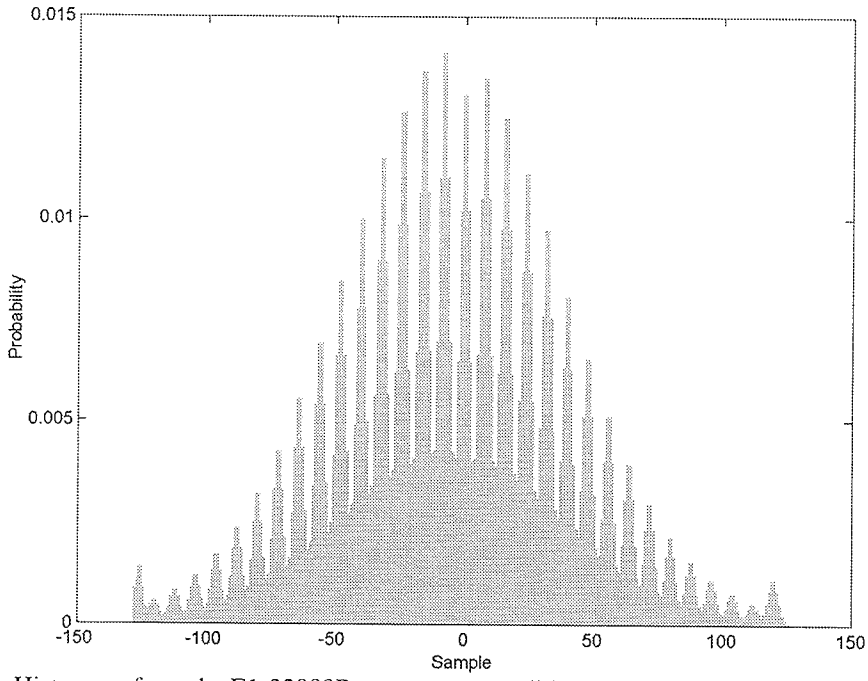


Fig. 5.2. Histogram from the E1-22089R test set pre-conditioned using the white noise technique.

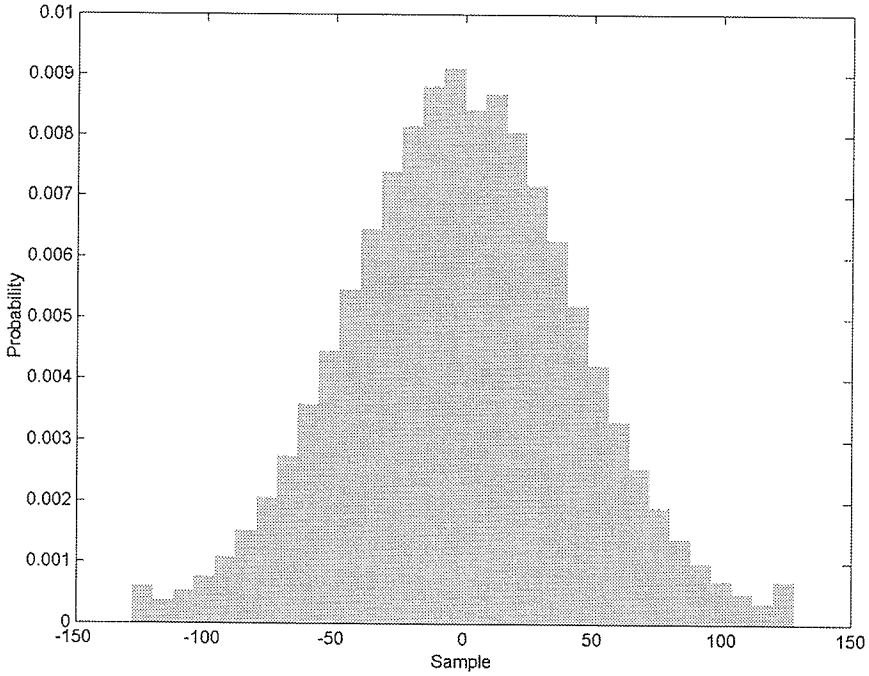


Fig. 5.3. Histogram from the E1-22089R test set pre-conditioned using the uniform noise technique.

The data set produced using uniform noise, which although looks acceptable, has very large saturation tails. Recall that one of the requirements was that the frequency of saturation be less for the pre-conditioned data than for the original data. Rather than lowering this saturation, the uniform noise technique has merely spread it out.

To address this problem of saturation at the tails, [Dutk93] reduces the saturation level by reducing the data range and adding negative exponential noise at the tails, and uniform noise elsewhere. The MDA pre-conditioning technique produces data with histograms similar to the one shown in Fig. 5.4. The MDA technique can be shown to produce data that satisfies all statistical requirements [Dutk93], but the histogram is not as smooth as would be desired.

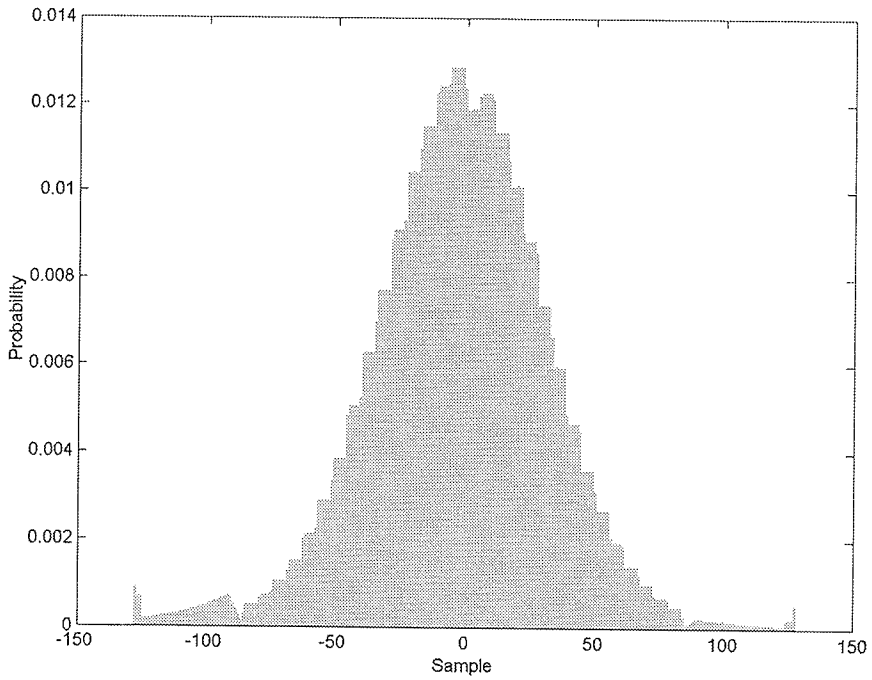


Fig. 5.4. Histogram from the E1-22089R test set using the MDA pre-conditioning technique.

In an attempt to improve upon the MDA technique a *cubic spline interpolation* [PrFT92] was investigated. Using this technique, each range line is in-turn fitted using a cubic spline and is then evaluated using an offset. Following this interpolation, the data is normalized and requantized to the desired number of bits. The resulting histogram is shown in Fig. 5.5. As can be seen from the figure, the shape of the histogram is very smooth, and the saturation at the tails is less than the original data. There are, however, small local regions of higher than expected probability close to the tails as a result of the tail saturation of the original data.

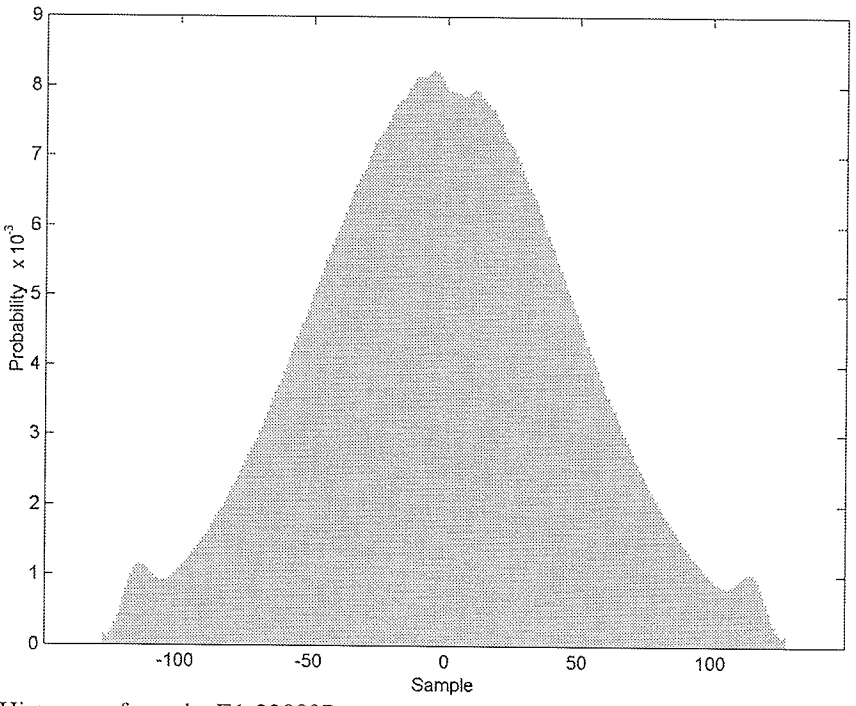


Fig. 5.5. Histogram from the E1-22089R test set pre-conditioned using the cubic spline technique.

Upon examination of the other statistical requirements of a data pre-conditioning technique, it can be seen that the cubic spline interpolation produces good results.

Statistics of the E1-22089R test set and resulting pre-conditioned data is shown in Table 5.2 with statistics for all data sets given in Appendix B.

Table 5.2 Statistics of the original and cubic spline pre-conditioned data from the E1-22089R test set.

Statistic	Original	Pre-conditioned
Entropy	4.52	7.47
Redundancy	0.48	0.53
Standard deviation	16.37	133.68
σ factor		8.17
Kurtosis	1.44	1.39
Skewness	1.16	1.14
Mean	0.001	0.005

As seen in Table 5.2, the entropy difference is very similar, with the entropy of the pre-conditioned data only slightly less. The other statistics are also all in line with expectations, being similar or off by a factor, such as the standard deviation. This standard deviation factor occurs as a result of the data expansion to 8 bits and is 2^{α}, where α is 8 minus the original number of bits. The cubic spline technique for the creation of pre-conditioned test sets can be broken down into the following four steps as shown in Table 5.3.

Table 5.3 Cubic spline based pre-conditioning test set generation algorithm.

Step	Description
1	Read a range line from the original raw SAR data test file.
2	Subtract the mean from the data line.
3	Fit a cubic spline to the data line.
4	Interpolate the spline at an offset ε to the right of the data point.
5	Add the minimum and divide by the maximum of the interpolated data line so that it lies between [0, 1].
6	Quantize the interpolated data line using a uniform quantizer to the desired number of bits.

This cubic spline technique is the technique used to generate the pre-conditioned test sets used in this thesis. Additional information about the test sets can be found in Appendix B, and MatLab source code for the cubic spline technique can be found in Appendix C.

5.3 Wavelet MRA Algorithm

As discussed in Chapter 4, a DWT analysis can be performed using MRA according to the equations

$$a^j{}_n = \sum_{k \in Z} \tilde{h}[2n-k]a_n^{j-1} \tag{5.1}$$

$$d^j{}_n = \sum_{k \in Z} \tilde{g}[2n-k]a_n^{j-1} \tag{5.2}$$

where a_n^j and d_n^j are the n th approximation and detail coefficients respectively at the resolution j and the filters $\tilde{h}[n]$ and $\tilde{g}[n]$ are specified for a given wavelet. The MRA analysis algorithm is therefore simply a convolution between the filter and the coefficients of the previous resolution followed by a downsampling by 2 as shown in Fig. 5.6.

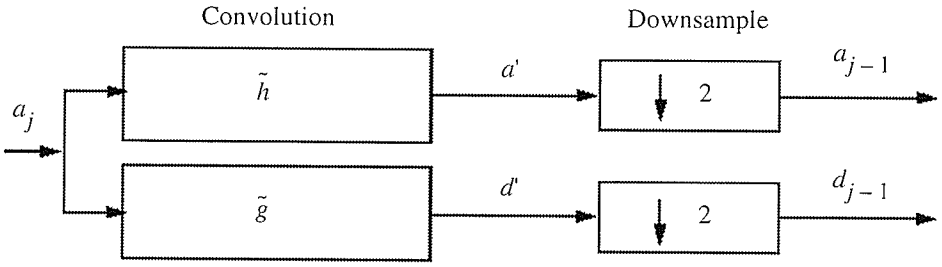


Fig. 5.6. Block diagram of MRA analysis algorithm (From [MMOP02]).

5.3.1 Implementation of the MRA Algorithm

For any practical wavelet filter of length l, the convolution for the n th coefficient of Eq. 5.1 becomes

$$a'_n = \sum_{k=0}^{l} \tilde{h}[n-k] a_n^{j-1} \tag{5.3}$$

Initially, this is a straight forward summation of l multiplications for each a'_n as the filter is defined to have l coefficients and be zero elsewhere, as shown in Fig. 5.7.

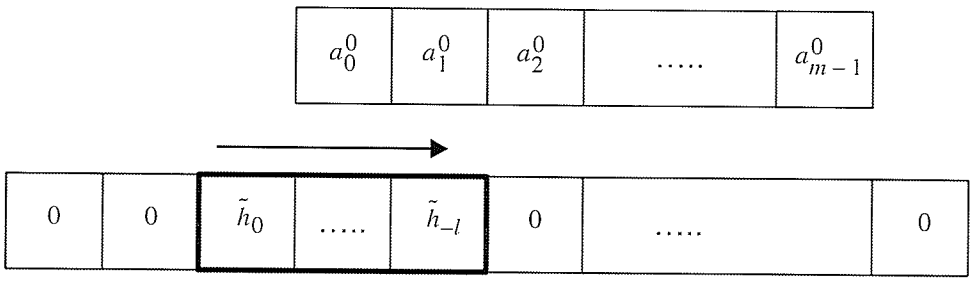


Fig. 5.7. Convolution of $\tilde{h}[n]$ with A_0.

To illustrate the convolution, consider the Haar wavelet filter with coefficients defined as

$$\tilde{h}_{-1} = \frac{1}{\sqrt{2}}, \tilde{h}_0 = \frac{1}{\sqrt{2}} \tag{5.4}$$

and the resulting coefficients of the convolution with a sequence A_0 of length m are:

$$\begin{aligned} a'_0 &= \tilde{h}_0 a_0^0 + \tilde{h}_{-1} a_1^0 \\ a'_1 &= \tilde{h}_0 a_1^0 + \tilde{h}_{-1} a_2^0 \\ &\cdots \\ a'_{m-1} &= \tilde{h}_0 a_{m-1}^0 + \tilde{h}_{-1} a_m^0 \end{aligned} \tag{5.5}$$

The question then becomes, if A_0 is of length m, having coefficients a_0^0 to a_{m-1}^0, what is the value of a_m^0. This problem, which is also seen in the MRA synthesis algorithm, is a practical problem of defining what the values of convolution sequence are outside its defined range.

Investigation into this issue results in consulting the properties of the discrete convolution theorem [GoWo92], which states that for a discrete convolution to be consistent with periodicity properties of the Fourier transform the discrete function to be convolved is assumed periodic with period M. In the literature [StNg96], [MMOP02], there have been several approaches for making A_j periodic including zero-padding, smooth padding, periodic extension, boundary value replication, and infinite period, each providing different effects to the resulting coefficients on the boundaries.

This thesis will use the periodic extension of A_j, as done in the Rice Wavelet Toolbox [BCFH00]. This involves replicating all the coefficients of A_j at m intervals, as shown in Fig. 5.8. A property of using periodic extension in the MRA algorithm is that for a signal with an even number of coefficients n, the resulting MRA analysis will produce exactly $n/2$ approximation coefficients and $n/2$ detail, and only these coefficients are needed for a perfect reconstruction. Note that this is not always the case for different periodic schemes.

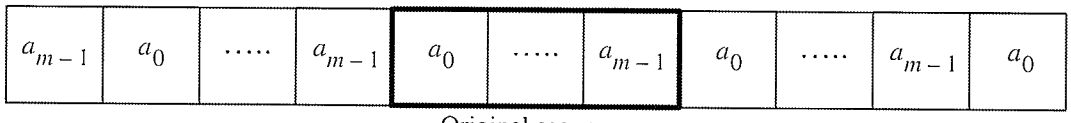


Fig. 5.8. Periodization of A.

Once the convolution of A_j with $\tilde{h}[n]$, or $\tilde{g}[n]$ has been performed, the next step is the downsample by 2. This step simply involves picking only every second coefficient after the convolution starting from 0, which is only the even coefficients. The MatLab code of the MRA algorithm used in this thesis is listed in Appendix C.

5.4 Quantization of the MRA Coefficients

Once the MRA algorithm has been applied to the raw SAR data, the resulting subbands must be quantized for transmission. If the distribution of the resulting subbands is known in advance, an optimum quantizer can be constructed to exploit this information.

For this thesis, the raw SAR data will only be decomposed to resolution 1. This choice was made to simply limit the number of variables analysed in this thesis, leaving a determination of the optimum decomposition resolution for future work. This choice then requires the construction of only two quantizers, one for the A_1 signal and one for the D_1 signal. It has been shown in previous work [BuSi99] that the MRA subbands statistics of natural images are the statistics of the original image multiplied by a stochastic process determined by the wavelet. This also appears to be the case for SAR by computing the histograms and normal probability plots of the original signal to the A_1 and D_1 signals as shown in Fig. 5.9 to 5.13 for the first 1000

samples of the E1-22089PS test set. Complete figures of all test sets are shown in Appendix E.

The mean of two MRA subbands was also calculated for the five test sets and is shown in Table 5.4. The data, histograms, and normal plots of the subbands shows that they are not exactly zero-mean Gaussian distributed, but that they are very close. With the observed statistics of the A_1 and D_1 signals, a Gaussian PDF optimized quantizer can be chosen and will be suitable for the quantization of the MRA coefficients. Note that this decision is only discussed for the case of descending to MRA resolution 1, and may not be valid at other resolutions.

Table 5.4 Mean of the A_1 and D_1 MRA subbands for the first 10000 samples of the E1-25224PS, E2-5551PS, R1-24919PS and R1-24576PS test sets using the db-2 wavelet.

Test set	Mean	
	A_1	D_1
E1-22089PS	-0.381	0.208
E1-25224PS	-3.143	-1.102
E2-5551PS	2.468	-0.119
R1-24919PS	-2.491	-0.119
R1-24576PS	1.909	-0.192

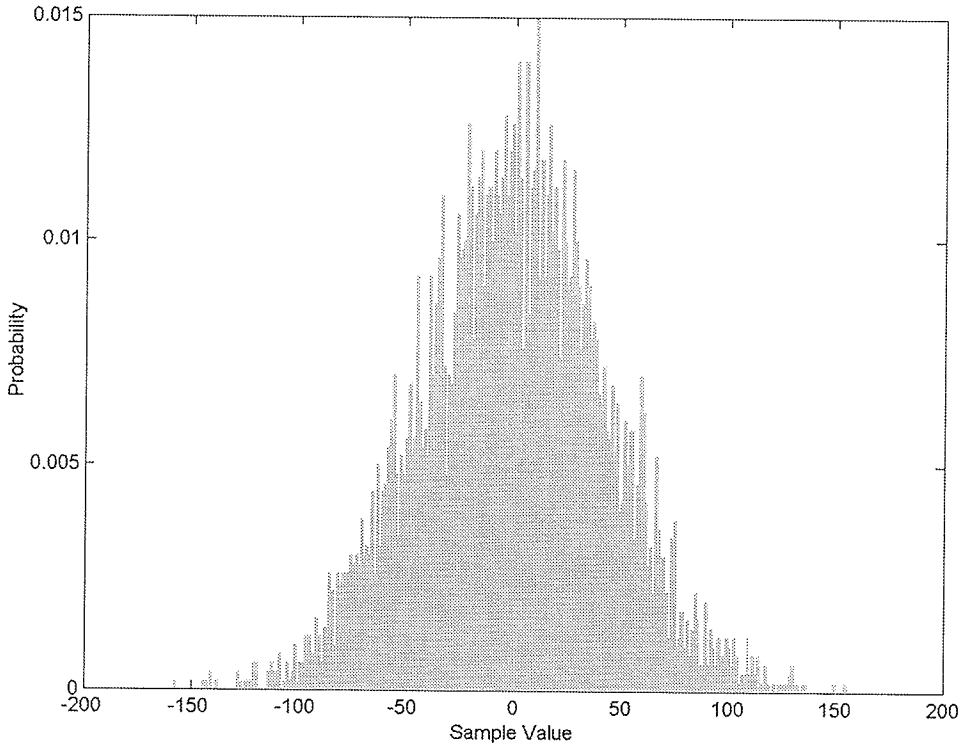


Fig. 5.9. Histogram of A_1 from first 10000 samples of E1-22089PS test set using the db-2 wavelet.

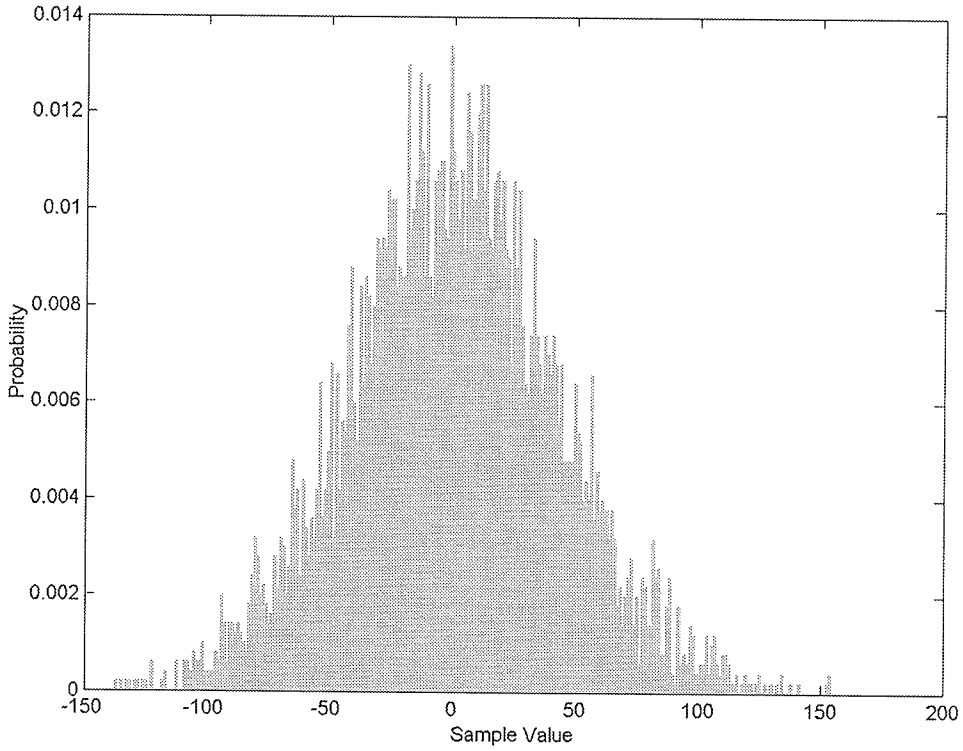


Fig. 5.10. Histogram of D_1 from first 10000 samples of E1-22089PS test set using the db-2 wavelet.

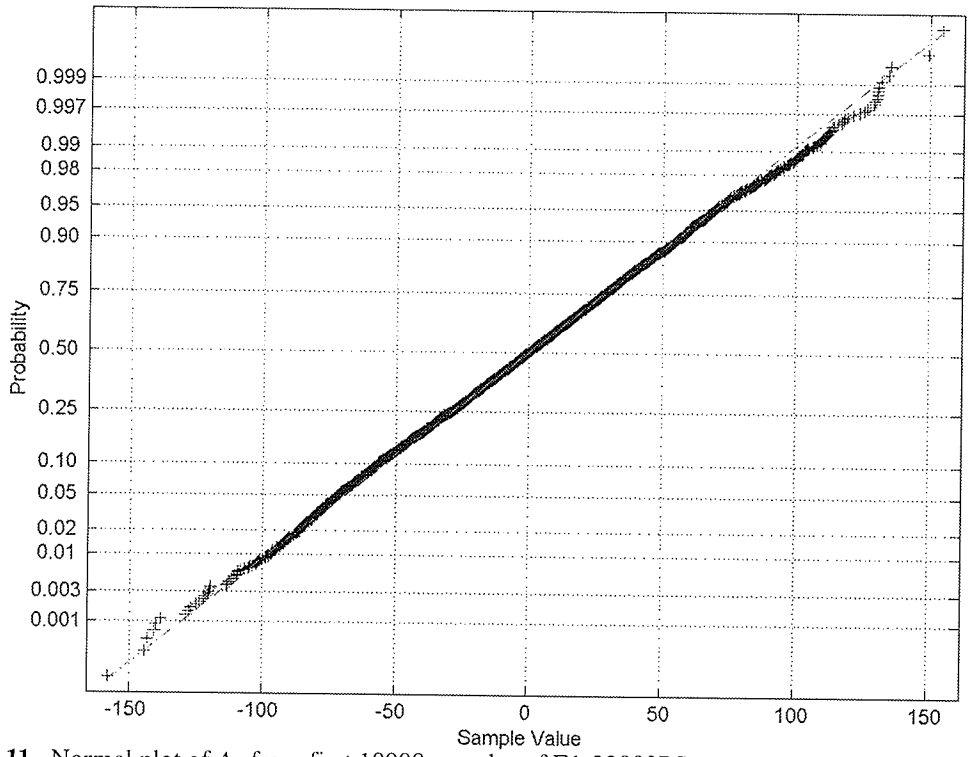


Fig. 5.11. Normal plot of A_1 from first 10000 samples of E1-22089PS test set using the db-2 wavelet.

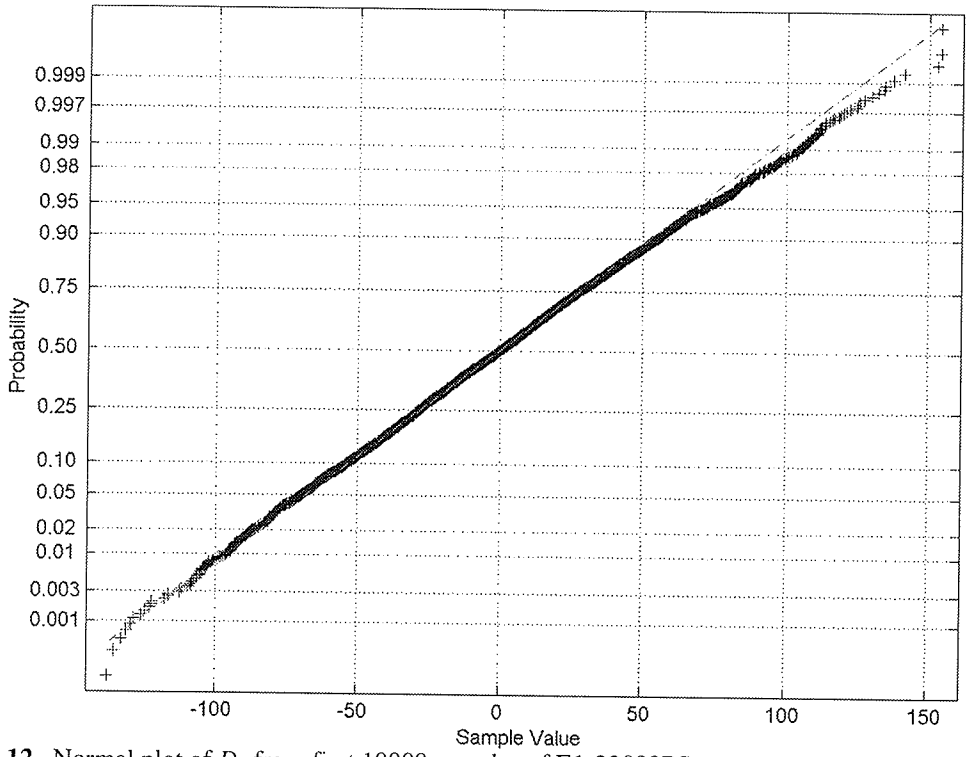


Fig. 5.12. Normal plot of D_1 from first 10000 samples of E1-22089PS test set using the db-2 wavelet.

5.4.1 Bit Allocation

The fundamental idea behind data compression using wavelets is the MRA analysis of the data using a wavelet ψ to a resolution j by recursively applying Eq. 5.1 and Eq. 5.2, followed by a quantization of the coefficients. As in other transform based compression schemes the bit allocation for the coefficients must be determined.

The problem of bit allocation is to determine the optimum number of bits to assign to each coefficient R_k, subject to a given quota R, to minimize the overall distortion D. The bit allocation equation used in this thesis is derived using high-resolution approximations and is given as [GeGr91]

$$R_k = R + \frac{1}{2}\log_2 \frac{\sigma_k^2}{\prod_{k=1}^{j+1} (\sigma_k^2)^{\frac{1}{j+1}}} \tag{5.6}$$

where σ_k^2 is the variance of the kth coefficient.

When applying Eq. 5.6 to the bit allocation of wavelet coefficients, the bit allocation is calculated for each of the subbands thus creating a separate bit allocation for each of the detail signals D_1 to D_j and the approximation signal A_j. This differs from the bit allocation from other transforms, such as the DCT, where a bit allocation is calculated for each of the coefficients.

For this work, the raw SAR data will only be decomposed to resolution 1, thereby creating only two signals A_1 and D_1, and only two bit allocations. This choice is made to reduce the number of variables examined in this thesis and the complexity of the hardware implementation. Investigation into the optimum MRA resolution is left for future work. With only two subbands, the bit allocation of A_1 using Eq. 5.6 is reduced to

$$R_{A_1} = R + \frac{1}{2}\log_2 \frac{\sigma_{A_1}^2}{\sqrt{\sigma_{A_1}^2}\sqrt{\sigma_{D_1}^2}} \tag{5.7}$$

Equation 5.7 shows that if the bit allocation of the approximation signal A_1 is to be one bit larger than the average bits per symbol R, then the variance of A_1 must be 16 times larger than the variance of D_1. Experimental results applying Eq. 5.7 to the first 400 blocks of the E1-22089PS, E1-25224PS, E2-555PS, R1-24919PS and R1-24576PS test sets using the Haar, Daubechies-4, Symlets-4, and Biorthogional 6.8 wavelets show that there is very little variation from the average bits per symbol rate R.The bit allocation for the E1-22089PS dataset using Eq. 5.6 and $R = 2$ is shown in Fig. 5.13 with complete figures of all test sets shown in Appendix E.

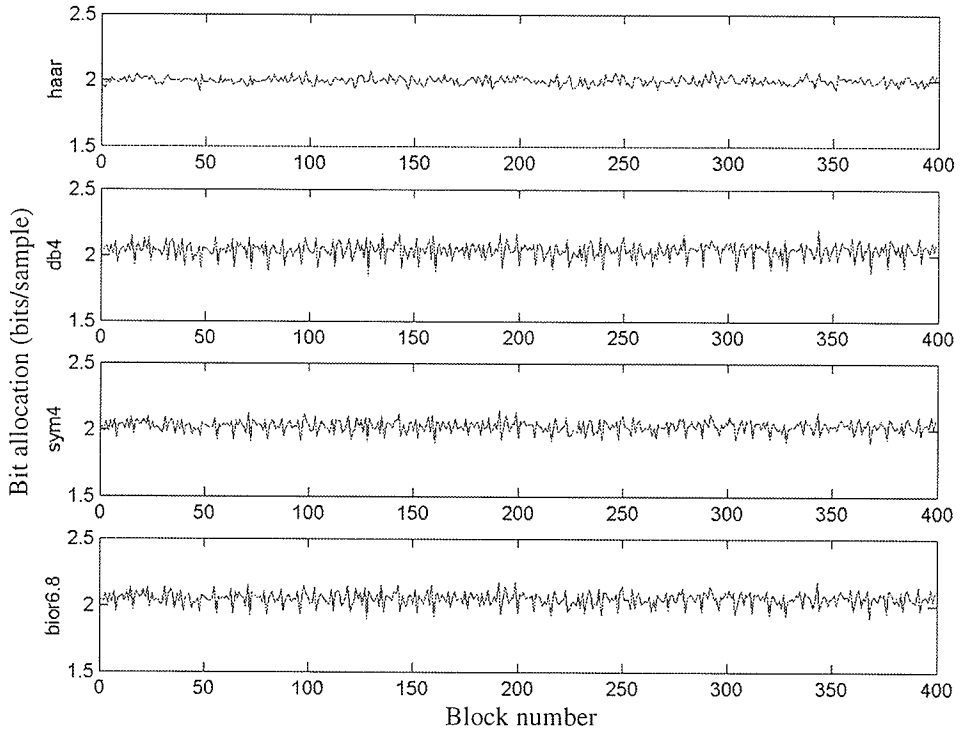


Fig. 5.13. A_1 bit allocation from the E1-22089PS test set the Haar, db4, sym4, and bior6.8 wavelets.

An interesting feature of the bit allocation results is that the shape of the plot is noticeably different for the Radarsat and ERS radars. This suggests that the design of a wavelet based technique could be optimized to the specific radar performing the imaging. The optimization of the technique for a specific radar is outside the scope of this thesis and is left for future work. For all test sets examined, the variation of the bit allocation around R is small and is due to the small ratio of $\sigma_{A_1}^2 / \sigma_{D_1}^2$ as shown in Fig. 5.14 for the E1-22089PS test set with complete figures of all test sets shown in Appendix E.

The small $\sigma_{A_1}^2/\sigma_{D_1}^2$ ratio observed in all the test sets allows for the decision of assigning R bits per sample for coefficients from both A_1 and D_1, thus removing any adaptive bit allocation. Note that this is only for the case of descending to MRA resolution 1, and may not hold true for other resolutions.

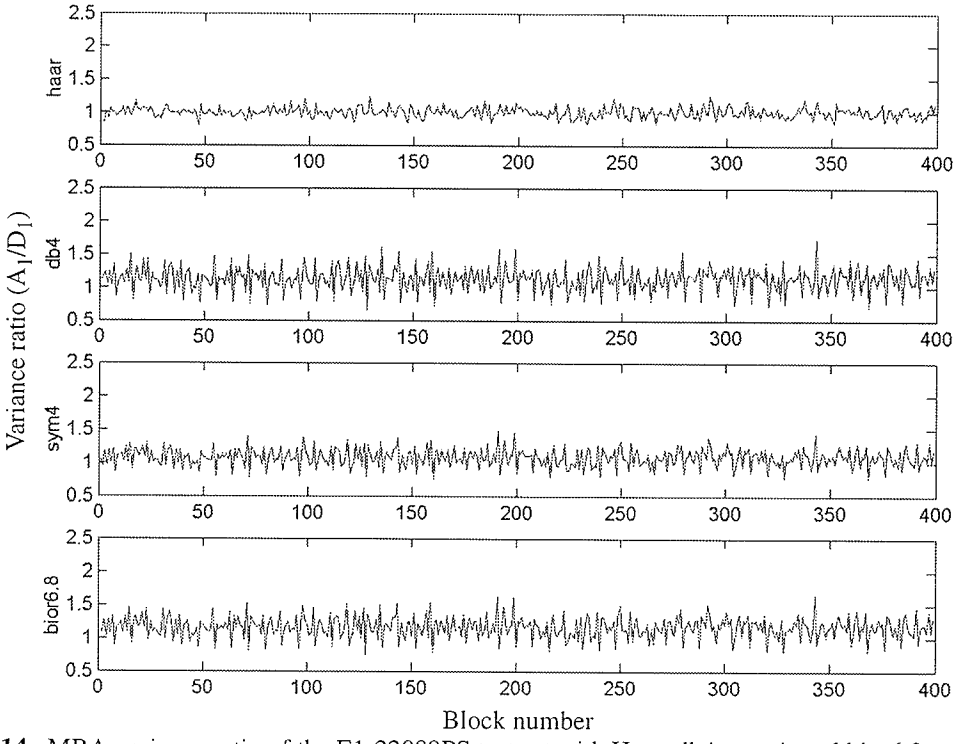


Fig. 5.14. MRA variance ratio of the E1-22089PS test set with Haar, db4, sym4, and bior6.8 wavelets.

5.4.2 Block Size

The two MRA subbands have been shown to be approximately zero-mean Gaussian distributed signals, with non-identical variances. The raw SAR data has slowly changing variance in both azimuth and range directions, which also indicates that the A_1 and D_1 subbands will have slowly changing variance. To effectively

quantize the MRA coefficients, an adaptive quantizer for zero-mean Gaussian distributed data is needed for each subband. Fortunately, one has already been developed, namely BAQ. The BAQ algorithm is applied on blocks of raw SAR data, or in this case, blocks of MRA coefficients, so the block size and dimensions must be determined. When BAQ is combined with the MRA algorithm on raw SAR data, the resulting technique is a wavelet-BAQ.

The first question is, should a 1-D or 2-D block of data be used, and if 1-D then what direction. To minimize the buffering needed, the BAQ, and therefore the MRA, used in this thesis will only operate on 1-D block of data in range. Because each range line of data corresponds to the radar return from one radar pulse, any block that would include multiple samples in azimuth would require buffering the previous range lines until the number of azimuth samples is available. The memory requirements on such a technique would be quite large as each range line contains over 10000 samples. Processing the data in range is equivalent to processing it as it is digitized by the A/D converter. The resulting wavelet-BAQ technique operating in range is a 1-D wavelet-BAQ.

The block size for the quantization, just as for BAQ, is a balancing of parameters between having adequate samples for a valid estimation of the standard deviation needed for the and quantizer and making sure the statistics of the block are stationary. Typical block sizes are 512 to 4096 with 1024 being the most common The block sizes for each of the test sets used in this thesis are shown in Table 5.2 and are calculated as

the closest multiple of the number of samples per range line to 1024. More detailed information about the test sets used in this thesis are given in Appendix B.

Table 5.5 Block size for pre-conditioned test sets.

Test set	Complex samples per range line	Total samples per range line	Quantization block size
E1-22089PS	5544	11088	1386
E1-25224PS	5616	11232	1404
E2-5551PS	5616	11232	1404
R1-24576PS	6708	13416	1677
R1-24919PS	6708	13416	1677

Some previous transform based techniques for raw SAR data compression have first normalized the block of data to be compressed before applying the transform [BeSM95], [MaOP02]. The logic behind the normalization is that the quantizers required to quantize the transform coefficients will not have to be adaptive as the variance of the coefficients will always be the same. This approach for wavelets, which although simplifies the quantizer required to quantize the coefficients, essentially requires three quantizers. One quantizer, a BAQ-8 or similar, would be required to normalize the raw SAR data, and two other quantizers would be required to quantize the approximation and detail coefficients. A pre-normalization scheme has the additional drawback of introducing more numerical error into the quantization technique in any practical implementation due to the limited accuracy of the outputs from the normalizer.

The quantization scheme for the coefficients in this work will not normalize the block of data prior to transformation, but will instead use two adaptive quantizers to quantize the MRA coefficients. This can be accomplished because the transform coefficients have approximately the same statistics as the original signal, namely zero mean Gaussian with slowly changing, but different, variance. The quantizers necessary for the coefficients are then BAQs where the quantizers are adapted to the variance of the coefficient blocks.

5.5 Optimum Standard Wavelet Basis

Up to this point, the wavelet-BAQ algorithm has been described, except for one important parameter, the wavelet itself. To determine the best wavelet the 1-D wavelet-BAQ algorithm described in the previous section is programmed in MatLab and is run on all five pre-conditioned raw SAR data test sets using the SQNR and quantization gain as metrics. The SQNR provides a solid qualitative assessment of the impact due to the data quantization, and the gain of the quantization gives an evaluation on the radiometric distortion [DuCu94] The 1-D wavelet-BAQ will also be run using 2, 3, and 4 bits per sample quantization and use the BAQ block sizes from Table 5.5.

To apply a comparison for the 1-D wavelet-BAQ compression technique, the standard technique of BAQ is also programmed in MatLab. The SQNR and quantization gain results is obtained using BAQ for each of the test sets and each

desired data rate. Using the MatLab 1-D wavelet-BAQ technique, the SQNR and quantization gain for all standard wavelet basis function in MatLab is calculated. The top five wavelet basis functions giving the best average SQNR on the first 200 blocks of the five test sets is shown in Table 5.6 compared to BAQ and the Shannon bound. Complete results for all standard wavelet basis functions for all test sets is shown in Appendix E, and MatLab code for the MatLab 1-D wavelet-BAQ algorithm is listed in Appendix C.

Table 5.6 SQNR the top 5 wavelet basis from the first 200 blocks of the E1-22089PS and R1-24576PS test sets compared to BAQ and the Shannon bound.

Wavelet basis	Mean SQNR in dB		
	2 bit/sample	3 bit/sample	4 bit/sample
Shannon bound	12.04	18.06	24.08
BAQ	9.52	15.00	20.42
Wavelet mean	9.17	14.49	20.17
db2	9.34	14.69	20.40
sym2	9.34	14.69	20.40
coif1	9.33	14.69	20.40
haar	9.33	14.67	20.39
db3	9.32	14.65	20.37

As shown from the results in Table 5.6 and Appendix E, there is not a substantial difference between the different standard wavelets. The mean SQNR of the wavelets for 2 bits/sample is 9.17 dB and there is a standard deviation of 0.30. The mean SQNR

for 2 bits/sample is shown graphically in Fig. 5.15, with only Daubechies wavelets shown in Fig. 5.16.

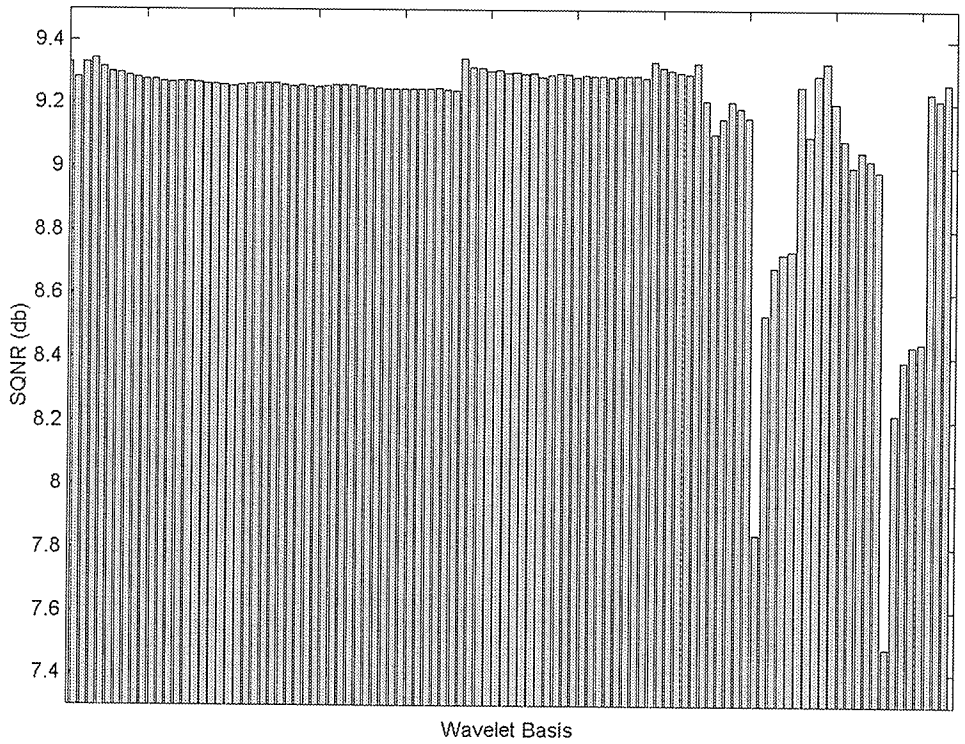


Fig. 5.15. Average SQNR of wavelet-BAQ using 2 bit/sample quantization and all standard wavelets.

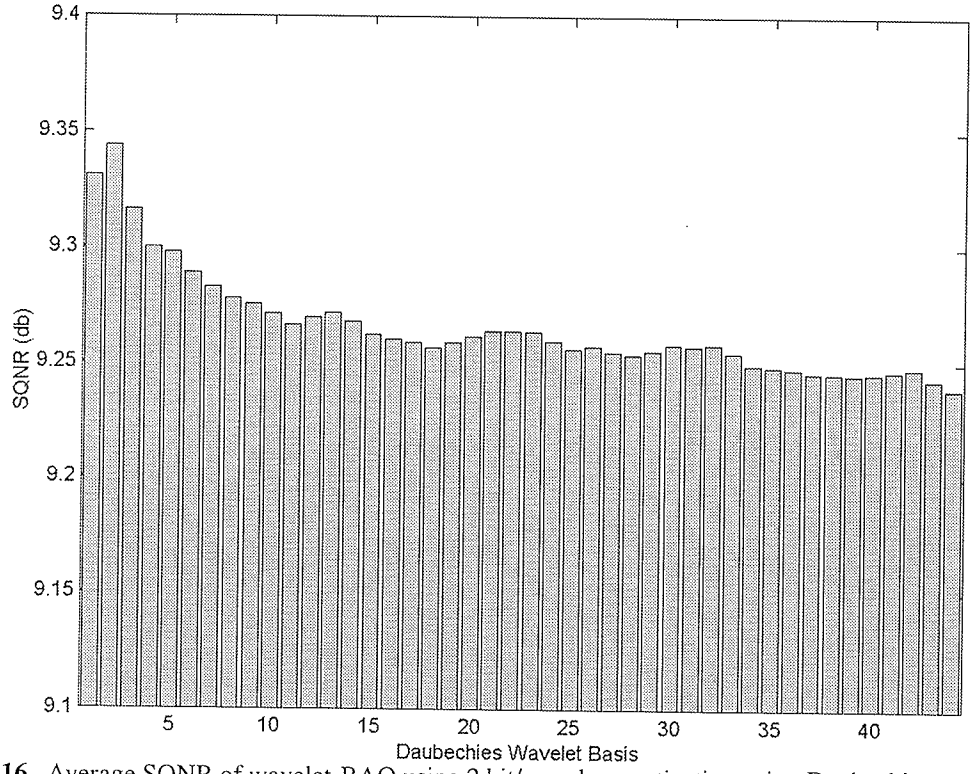


Fig. 5.16. Average SQNR of wavelet-BAQ using 2 bit/sample quantization using Daubechies wavelets.

The results of all the standard wavelet basis in MatLab reveal an interesting observation about the wavelet compression of raw SAR data. Specifically no standard wavelet function is adequately similar to the raw SAR signal. This can be seen from the SQNR performance of the wavelets which are consistently less than that of BAQ for all test sets and all bit rates reflecting the fact that no wavelet is adequately similar to the SAR signal.Although no standard wavelet basis functions used in this thesis have produced SQNR performance greater than BAQ, a custom wavelet basis function could be created that could be placed directly into the 1-D wavelet-BAQ algorithm. The possibility of such a wavelet validates the need for a hardware implementation of the 1-D wavelet-BAQ algorithm as once the wavelet is created, and corresponds to a MRA, the wavelet coefficients could then be placed directly into the 1-D

wavelet-BAQ implementation. The development of a custom wavelet for the compression of raw SAR data is left for future work. The 1-D wavelet-BAQ technique used to compress raw SAR data for this thesis can be broken down into the following four steps as shown in Table 5.7, with MatLab code listed in Appendix E.

Table 5.7 1-D wavelet-BAQ algorithm for raw SAR data compression.

Step	Description
1	Acquire a 1-D block of raw SAR data in range.
2	Decompose the block into an approximation signal A_1 and detail signal D_1 using MRA and the Daubechies 2 wavelet.
2	Estimate the variance of A_1 and D_1.
3	Use the variance of the coefficients to determine the optimum thresholds for the Max-Lloyd quantizers.
4	Compress the coefficients using separate Max-Lloyd quantizers and a common bits per sample R.

5.6 Summary

It is the aim of this thesis to develop a useful wavelet based compression scheme for raw SAR data that is low enough in computational complexity to make it a candidate for on-board data compression. Achieving this goal requires that the technique provide potentially better performance results than the standard BAQ technique, but also not require excessive computation.

Using the recursive MRA algorithm developed by Mallat, the MRA technique for wavelet decomposition can be implemented by a convolution followed by a

downsampling. Several parameter decisions specific for the MRA technique used in this thesis have been described in this chapter. The MRA will only descend to resolution 1 to produce the A_1 and D_1 signals, where the block size used for A_0 is a function of the dataset file.

After examining the variance of the A_1 and D_1 signals for various wavelets, it was found that no adaptive, or different, bit allocation technique was required as the optimum bit allocation was always around R. This was found to be a result of low small ratio of $\sigma_{A_1}^2 / \sigma_{D_1}^2$, and allows the bit allocation for both A_1 and D_1 to be R. An examination of the statistics of the A_1 and D_1 signals also shows that their statistics can be modelled as zero-mean Gaussian with slowly changing signal power. The BAQ algorithm can then be used to quantize the A_1 and D_1 signals.

The choice of 1-D or 2-D block sizes was chosen to be 1-D in range as this will remove the need for any additional buffering, and will simplify the wavelet-BAQ algorithm. This choice also implies a technique that processes the SAR data directly as it leaves the A/D converter.

The final parameter that was left unknown, was what wavelet to use. After experimenting with all standard wavelet basis in MatLab it was discovered that no wavelet performed significantly better than any other, and that no wavelets performed better than BAQ. Although no wavelet was superior to BAQ, the wavelet-BAQ

algorithm has the potential to outperform BAQ if a wavelet is found to be better suited to the SAR signal. To implement the wavelet-BAQ algorithm, the Daubechies-2 wavelet was chosen as it showed the highest average SQNR in the experiments. With all aspects of the 1-D wavelet-BAQ algorithm discussed, the algorithm can now be implemented in hardware.

CHAPTER 6

IMPLEMENTATION OF THE 1-D WAVELET-BAQ TECHNIQUE

The 1-D wavelet-BAQ algorithm developed in Chapter 5 has shown good SQNR performance results on the pre-conditioned test sets, but the true test for any on-board raw SAR data compression technique is whether or not it can be implemented in low complexity hardware. The algorithm described in Chapter 5 addressed all the practical issues with the MRA and BAQ such as the convolution, the wavelet, and the quantization of the coefficients and if the target of the hardware implementation was a floating-point processor, then the 1-D wavelet-BAQ algorithm could be directly implemented.

For this thesis, the desired hardware target is an integer only arithmetic custom computing engine (CCE) that could be implemented as a radiation hardened ASIC. This constraint on the hardware implementation dictates that no floating-point arithmetic can be used, and all calculations must be done on integers. A second constraint on the hardware is that buffering and memory should be kept to a minimum. Acknowledging these constraints requires the re-examination of the 1-D wavelet-BAQ for implementation using integer arithmetic only.

Two main issues of the 1-D wavelet-BAQ algorithm will be explored in this chapter before discussing any implementation. Firstly, how to calculate the standard

deviation, and therefore the quantizer thresholds, for a block of data without using a floating-point unit to calculate the standard deviation and without buffering the entire block before hand. Secondly, how to calculate the MRA coefficients using integer only arithmetic since the db-2 filter coefficients are all rational numbers.

This chapter presents the design and FPGA implementation of the 1-D wavelet-BAQ algorithm described in Chapter 5 as a CCE using integer-only arithmetic. To provide a performance measurement of the 1-D wavelet-BAQ, the BAQ algorithm is also implemented for comparison in Chapter 7.

6.1 Design of the Custom Computing Engine

The algorithm described in the previous chapter discussed a 1-D wavelet-BAQ for the compression of raw SAR data, as shown in Fig. 6.1, where the raw SAR data is decomposed to MRA resolution 1 using the db-2 wavelet and the A_1 and D_1 signals are quantized using a BAQ. This discussion detailed many of the practical implementation concerns of the algorithm, such as how to perform the MRA convolution, how to quantize the coefficients, and what wavelet to use. Although these specifications allowed the implementation of the 1-D wavelet-BAQ in MatLab, there are implementation parameters which must be considered for an integer CCE hardware implementation.

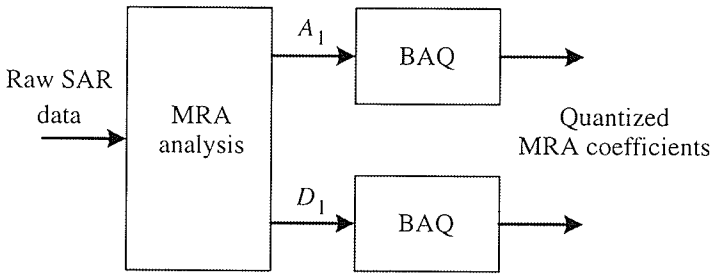


Fig. 6.1. Block diagram of wavelet-BAQ algorithm.

The end implementation for any on-board raw SAR data compression technique is the implementation as a radiation hardened ASIC, and the lower the complexity the smaller the ASIC size. Additionally, the less buffering required for the implementation, the cheaper the implementation. That being said, the implementation cannot be so simple that it requires enormous processing time or unacceptable performance results. The goal is then to implement the 1-D wavelet-BAQ algorithm as a low-complexity CCE that can run at frequencies greater than the throughput of the A/D converter, which for Radarsat-1 is 64.6 MS/s.

To implement the 1-D wavelet-BAQ in low complexity hardware, the algorithm needs to be re-examined and optimized for the target hardware. Several functions used in the MatLab implementation that used floating-point arithmetic will have to be converted to integer arithmetic before implementation. The specific procedures that will be examined are the calculation of the block variance and the quantization of the data of the BAQ, and the filter bank implementation of the MRA analysis algorithm.

6.1.1 Calculation of the Block Variance

The standard deviation σ of a zero mean Gaussian random variable x can be estimated from N points using

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} x_n^2} \tag{6.1}$$

and is all that is needed to completely describe the distribution. The standard deviation can then be used to calculate the optimal quantizer boundaries using the Max-Lloyd quantizer. Although Eq. 6.1 is easily calculated using a standard PC with floating-point capabilities, it is not as simple for a low complexity integer ASIC. The two problems are the large amount of multiplications and the calculation of the square root.

As a solution to this problem, Kwok and Johnson [KwJo89] suggested using the average signal magnitude statistic $|\bar{X}|$ to determine the standard deviation, as opposed to estimating the standard deviation directly. The average signal magnitude is attractive as it is much simpler to compute than the standard deviation.

To find a relation between the average magnitude statistic $|\bar{X}|$ and the standard deviation σ of a Gaussian distribution, let us assume that they are related by a factor m such that

$$\sigma = m|\bar{X}| \tag{6.2}$$

and the average magnitude statistic is defined as

$$\begin{aligned} |\bar{X}| &= \int_{-\infty}^{\infty} |x| f_x(x) dx \\ &= 2\int_{0}^{\infty} x f_x x dx \end{aligned} \tag{6.3}$$

where $f_x(x)$ is the PDF of the Gaussian input defined as

$$f_x(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{x^2}{2\sigma^2}} \tag{6.4}$$

For a zero-mean Gaussian distribution, Eq. 6.3 is equal to

$$\begin{aligned} |\bar{X}| &= 2\int_{0}^{\infty} \frac{x}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}} dx \\ &= \sigma\sqrt{\frac{2}{\pi}} \end{aligned} \tag{6.5}$$

Combining Eq. 6.5 with Eq. 6.2 gives

$$\begin{aligned} m &= \frac{\sigma}{\sigma\sqrt{\frac{2}{\pi}}} \\ &= \sqrt{\frac{\pi}{2}} \end{aligned} \tag{6.6}$$

which is approximately equal to 1.25. Therefore

$$\sigma \approx 1.25|\bar{X}| \tag{6.7}$$

Using Eq. 6.7, the standard deviation of raw SAR data block can be estimated very accurately. To illustrate this, the error between Eq. 6.7 and Eq. 6.1 for the first 400 blocks of the E1-22089PS test set is shown in Fig. 6.2.

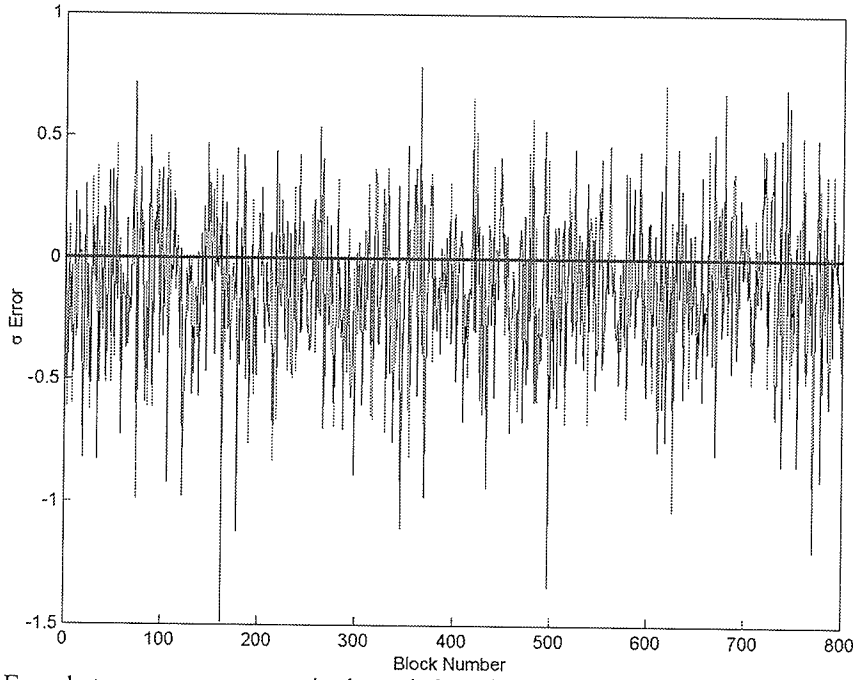


Fig. 6.2. Error between average magnitude static based estimation of standard deviation and direct estimation of standard deviation for first 400 blocks of the E1-22089PS test set.

A first reaction to Eq. 6.7 might be that it is still a floating point operation, as it is the product of a real number and an integer, and calculating $|\bar{X}|$ requires the division by the block size. This is of course true, however, Eq. 6.7 can easily be implemented

using a look-up table relating $|\bar{X}|$ to σ, and for a power of 2 block size $|\bar{X}|$ is simply the sum of all $|x|$ followed by a shift.

6.1.2 Max-Lloyd Quantization

To simplify the hardware implementation of this thesis, only a 2 bit/sample BAQ will be implemented with this simplification there are only three thresholds and four codewords for the quantizer as shown in Table 6.1.

Table 6.1 Intervals, codewords, and reconstruction values of a 2-bit Gaussian PDF quantizer.

Interval	Codeword	Reconstruction
$(-\infty, -0.9816\sigma)$	11	-1.5104σ
$[-0.9816\sigma, 0)$	10	-0.4528σ
$[0, 0.9816\sigma)$	00	0.4528σ
$[0.9816, \infty)$	01	1.5104σ

The quantizer is thus reduced to asking two questions of each piece of data x to be quantized, (i) is $x \geq 0$, and (ii) is $|x| > 0.9816\sigma$. Implementing these questions in hardware amounts to implementing two comparators, one comparing x to zero and the other comparing $|x|$ to 0.9816σ.

The comparison to zero is only a check of the most significant bit (MSB) of x, if x is a signed number, which can easily be implemented by simply transferring the MSB

of x to the MSB of the codeword. The only comparator needed is then the one comparing $|x|$ to 0.9816σ as shown in Fig. 6.3.

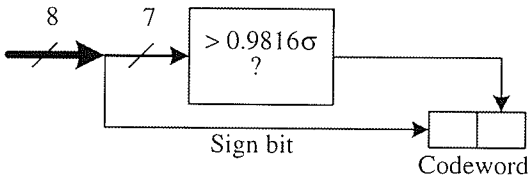


Fig. 6.3. Block diagram of Max-Lloyd quantizer.

The only parameter required for the quantizer is the standard deviation of the data block σ, which can be calculated using the average magnitude statistic described in Section 6.1.1.

6.1.3 MRA Analysis Using Polyphase Filter Banks

As previously discussed in Chapter 5, the DWT using MRA can be implemented by convolution with a filter and downsampling, as shown in Fig. 6.4.

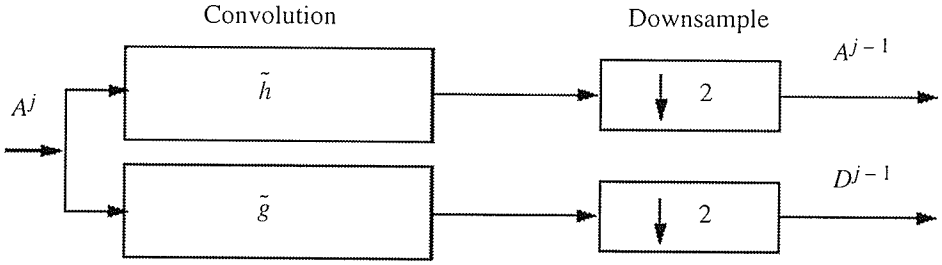


Fig. 6.4. Block diagram of MRA analysis algorithm (From [MMOP02]).

The convolution of the filter $\tilde{h}[n]$ with A^j in the MRA analysis algorithm can be expressed in the time domain for a filter of length N as

$$a^j[n] = \sum_{k=0}^{N-1} \tilde{h}[n-k]a^{j-1}[n] \tag{6.8}$$

and requires N multiplications and $N-1$ additions for each coefficient. The total computation required to convolve a signal of length M with two separate N length filters is then $2MN$ multiplications and $2M(N-1)$ additions. In the MRA analysis algorithm only every second coefficient is kept after the downsampling, resulting in a great waste of computation. An effective means of developing a convolution algorithm that computes only every second coefficient is the polyphase decomposition [Parh99] of the convolution. Two develop the algorithm, Eq. 6.8 can be expressed in the z-domain as

$$A^j[z] = \tilde{H}[z]A^{j-1}[z] = \sum_{k=0}^{N-1} \tilde{h}[k]z^{-k} \sum_{k=0}^{N-1} a^{j-1}[k]z^{-k} \tag{6.9}$$

The input sequence of $a^j[n]$ can be decomposed into two polyphases, one for the even indices and one for the odd indices as

$$\begin{aligned} A^j[z] &= a^j[0] + a^j[1]z^{-1} + a^j[2]z^{-2} + \ldots \\ &= a^j[0] + a^j[2]z^{-2} + a^j[4]z^{-4} + \ldots \\ &\quad + z^{-1}(a^j[1] + a^j[3]z^{-2} + a^j[5]z^{-4} + \ldots) \\ &= A^j_{even}[z^2] + z^{-1}A^j_{odd}[z^2] \end{aligned} \tag{6.10}$$

where $A^j_{odd}[z^2]$ and $A^j_{even}[z^2]$ are the z-transforms of $a^j[2k]$ and $a^j[2k+1]$ respectively. Likewise, the filter $\tilde{h}[n]$ can also be broken into two polyphases as

$$\tilde{H}[z] = \tilde{H}_{even}[z^2] + z^{-1}\tilde{H}_{odd}[z^2] \tag{6.11}$$

where both $\tilde{H}_{odd}[z^2]$ and $\tilde{H}_{even}[z^2]$ are of length $N/2$. Using the polyphases of the input and filter, the even index and odd index output sequence can be written as

$$\begin{aligned} A^{j-1}[z] &= A^{j-1}_{even}[z^2] + z^{-1}A^{j-1}_{odd}[z^2] \\ &= (A^j_{even}[z^2] + z^{-1}A^j_{odd}[z^2])(\tilde{H}_{even}[z^2] + z^{-1}\tilde{H}_{odd}[z^2]) \\ &= A^j_{even}[z^2]\tilde{H}_{even}[z^2] + z^{-1}A^j_{odd}[z^2]\tilde{H}_{even}[z^2] \\ &\quad + z^{-1}A^j_{even}[z^2]\tilde{H}_{odd}[z^2] + z^{-2}A^j_{odd}[z^2]\tilde{H}_{odd}[z^2] \end{aligned} \tag{6.12}$$

such that

$$A^{j-1}_{odd}[z^2] = A^j_{odd}[z^2]\tilde{H}_{even}[z^2] + A^j_{even}[z^2]\tilde{H}_{odd}[z^2] \tag{6.13}$$

and

$$A^{j-1}_{even}[z^2] = A^j_{odd}[z^2]\tilde{H}_{odd}[z^2] + z^{-2}A^j_{even}[z^2]\tilde{H}_{even}[z^2] \tag{6.14}$$

A block diagram of the polyphase convolution operation is shown in Fig. 6.5.

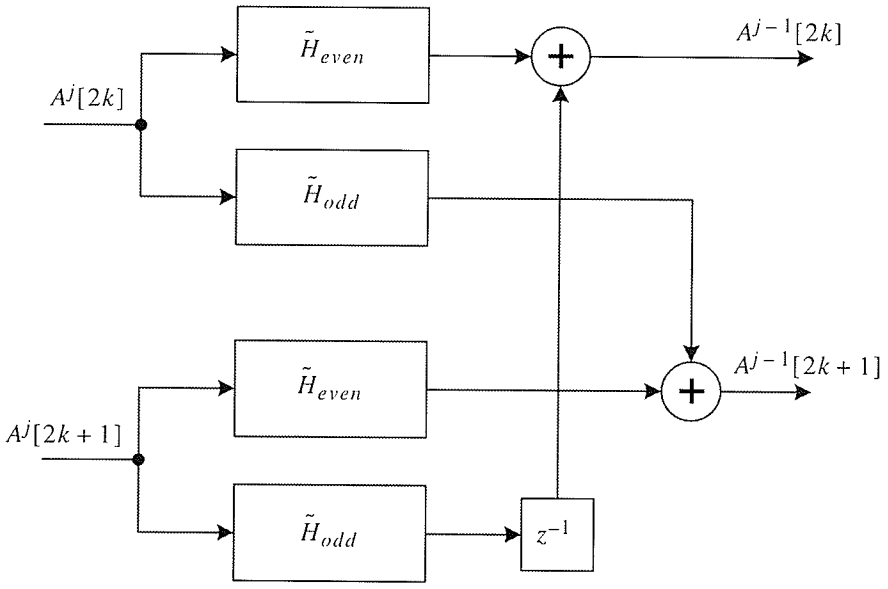


Fig. 6.5. Polyphase decomposition of FIR filter (After [Parh99]).

Recall that after the downsampling of the convolution, only the even index coefficients are kept, therefore the MRA analysis of A^j is

$$A^{j-1} = A^j_{even}[z^2]\tilde{H}_{even}[z^2] + z^{-2}A^j_{odd}[z^2]\tilde{H}_{odd}[z^2] \tag{6.15}$$

and

$$D^{j-1} = A^j_{even}[z^2]\tilde{G}_{even}[z^2] + z^{-2}A^j_{odd}[z^2]\tilde{G}_{odd}[z^2] \tag{6.16}$$

Using the polyphase decomposition, the total computation of the MRA analysis to decompose a signal of length M into the approximation and detail signal at the next resolution is then MN multiplications and $M(N-2)+2$ additions, a significant improvement. An additional advantage of the polyphase decomposition is that now 4

$N/2$ length filtering operating can occur at once, compared to just 2, speeding up the convolution. A block diagram of the polyphase MRA analysis is shown in Fig. 6.6.

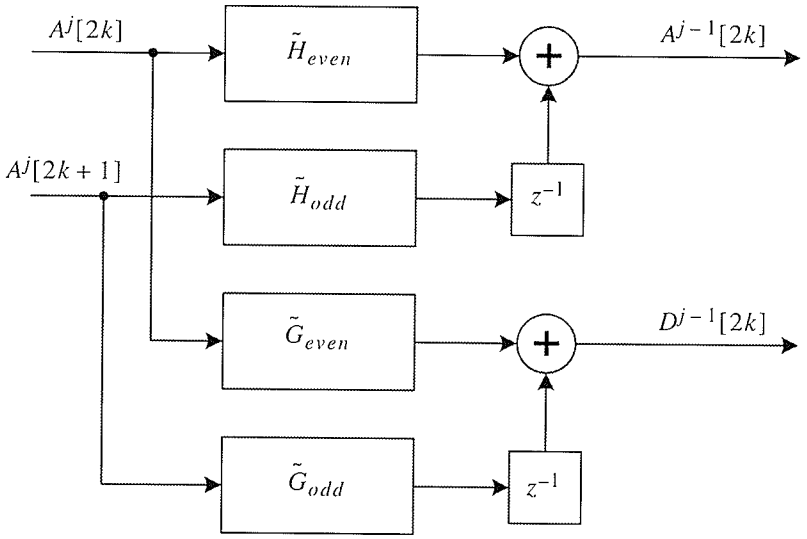


Fig. 6.6. Polyphase implementation of MRA algorithm.

In addition to decomposition into two polyphases, there are several other optimizations that can be applied to further optimize the filtering operation. Interested readers are directed to [Parh99] and [Meye01].

6.2 Implementation in an FPGA

The aim of this section is to discuss the implementation of the BAQ and 1-D wavelet-BAQ algorithms in FPGA hardware using the complexity reducing methods discussed in Section 6.1. The implementation should closely resemble an architecture for use on-board a satellite for raw SAR data compression. Unfortunately, a SAR

satellite development kit is not readily available, and as such the Ballynuey 2 PCI FPGA card by Nallatech was chosen to implement the FPGA designs.

6.2.1 Target System

The target hardware platform for the hardware implementations for this thesis is the Ballynuey 2 DIME PCI card developed by Nallatech incorporated [Nall01]. The Ballynuey 2 is a PCI carrier card populated with a dedicated PCI controller FPGA, a user FPGA, and up to 4 DIME modules. The system functions by programming the user FPGA, and DIME modules, from the host PC, then reading and writing information to the DIME system via the PCI bus. A block diagram of the DIME system is shown in Fig. 6.7.

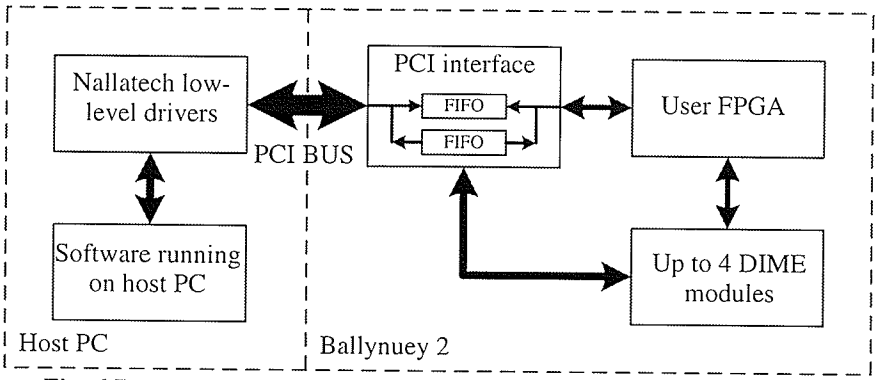


Fig. 6.7. Block diagram of Nallatech DIME card carrier (After [Nall01]).

The Ballynuey 2 system used in this thesis is populated with a Xilinx Virtex-600E FPGA as the user FPGA, and no additional DIME modules. The PCI FPGA is a Xilinx Spartan FPGA pre-configured by Nallatech and cannot be reprogrammed. To program the Virtex FPGA, a bitstream is first generated using the Xilinx toolchain, and the

FPGA is then programmed via PCI busing using the Nallatech low level drivers. Additional information about the Ballynuey 2 PCI card can be found in [Nall01].

From the Virtex FPGA point of view, the system is input data via a PCI interface bus of the PCI FPGA, and has access to 2 banks of 2 MB pipelined zero bus turnaround (ZBT) RAM [Bapa99], as shown in Fig. 6.8. The PCI interface FPGA handles all PCI communication and places the PCI to FPGA data in a FIFO. Likewise, all FPGA to PCI data is fed into a FIFO where is it read via the Nallatech low level drivers.

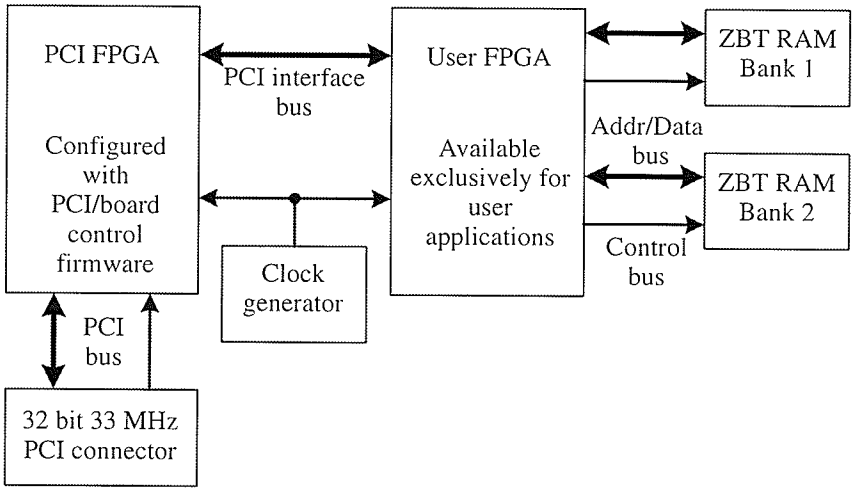


Fig. 6.8. Block diagram of Ballynuey 2 system from user FPGA perspective (After [Nall01]).

The ZBT RAM aboard the Ballynuey 2 are Late-Late Read RAMs [Bapa99] as during a read operation, valid data appears two cycles after addresses and control signals have been registered. Similarly, during a write operation, the data is input two clock cycles after the address and control signals are registered. A timing diagram of a read operation is shown in Fig. 6.9.

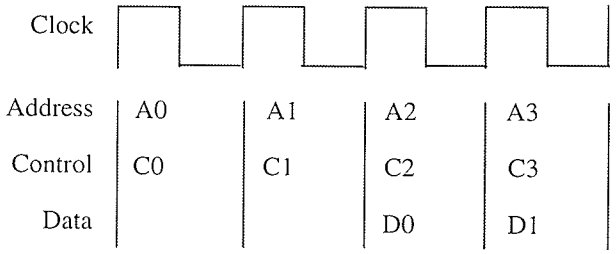


Fig. 6.9. Bus operations for read from pipelined ZBT SRAM (After [Bapa99]).

Using the Ballynuey 2 PCI card, an implementation resembling one that would appear on-board a SAR satellite, is desired. The basic SAR compression system is very simple, digitized SAR data enters the compressor and compressed data exits. The compressed data can then be transmitted to the ground station where the data is decompressed, or stored for later transmission. A block diagram of the on-board raw SAR data compression system is shown in Fig. 6.10.

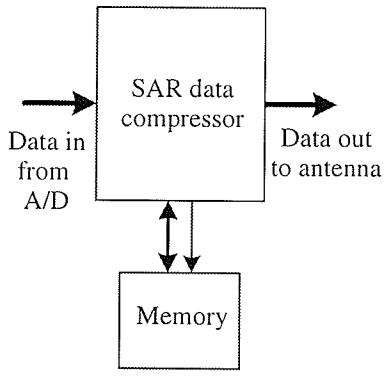


Fig. 6.10. On-board raw SAR data compression system block diagram.

The task is then to implement the raw SAR data compression system using the Ballynuey 2 PCI card. The input data and output data can come from/to the PCI bus,

and the SAR data compressor is simply a compression algorithm implemented using the user FPGA resources.

To develop a CCE that will read/write data to/from the PCI FPGA, a finite state machine (FSM) must be created to handle the communications protocol. The received data can then be processed by the user FPGA and sent back to the PCI FPGA for reading by the host PC. Although this scheme will function, and will implement the on-board raw data compression system model, a better implementation will separate the PCI FPGA communications FSM and the SAR compressor. The main reason for such a decision is that the PCI core FSM must run at 40 MHz to communicate with the PCI FPGA. By breaking the system into two parts, two clock domains can be used, one for the PCI FPGA communications and another for the SAR compressor. The interconnection between the two clock domains is then implemented using asynchronous FIFOs allowing buffering of data in both directions with minimal interfacing overhead. A block diagram illustrating the PCI FPGA communication FSM, called the PCI core, and the SAR compressor, called the user core, is shown in Fig. 6.11. This model is the one used for all implementations using the Ballanuey 2 in this thesis.

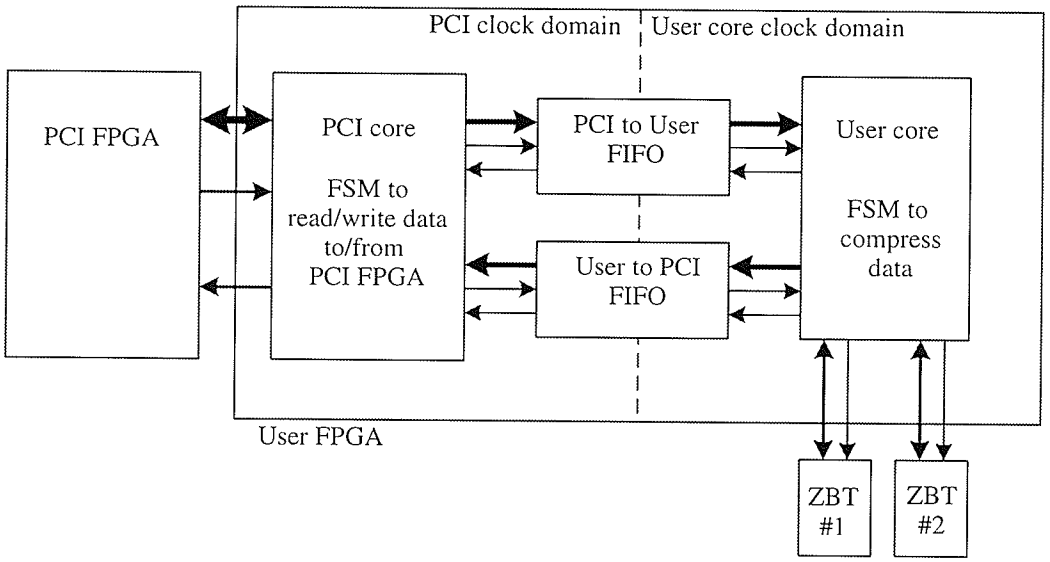


Fig. 6.11. Block diagram of the PCI core and user core.

The entire raw SAR data development system using the Ballynuey 2 PCI card works directly with the host PC. For this thesis, software was written using the Nallatech low-level drivers to write the SAR data test sets to the PCI FPGA, and read the quantized data from it. This software also reconstructed the received data creating a new data file representing the reconstructed data. This reconstruction step would be implemented as part of the SAR ground station, while the CCE would directly read data from the A/D converter aboard the satellite.

6.3 FPGA Implementation of BAQ

The FPGA implementation of BAQ implementation serves two purposes in this thesis. Firstly, it will provide an objective means of evaluating the performance of the new 1-D wavelet-BAQ as both will be implemented in the same environment.

Secondly, as the wavelet-BAQ requires a BAQ for the quantization of the MRA coefficients, any implementation of BAQ could be largely reused.

6.3.1 Implementation Considerations

The BAQ is simply a block based quantization scheme, where the quantizer is adapted to each block. By implementing the quantizer discussed in Section 6.1, the quantization is a simple comparison to the threshold 0.9816σ. Using this reduced complexity for the quantization, three issues are left. Firstly, the data from the A/D converter is from the interval [0, 255] and has a mean of 127.5, and not 0. Secondly, the determination of the quantization threshold using the average magnitude statistic. And last, how to calculate the necessary statistic without first buffering an entire block of data.

The first issue of removing the mean is not complex in appearance, however to avoid the use of floating-point numbers, some concessions are necessary. The choice is then what integer should be subtracted to make the samples zero mean, with the only logical choices being 128 and 127. Although both have the same magnitude error from the true mean, subtracting 128 will effectively change the sample to a 2's complement signed number.

The second and third issues are related in that they correspond to calculating the appropriate threshold for the block of range samples. In the MatLab implementation of

the BAQ algorithm, the standard deviation was estimated for each block as it was quantized. To do this in hardware would require the buffering of an entire block of samples before it could be quantized, which is not an attractive perspective as the desired implementation is to have minimal buffering. An alternate approach is to use the property of slowly changing variance in both range and azimuth directions of the raw SAR data by using the standard deviation of the previous block to calculate the quantization threshold. The first block in a scene to be compressed would then be assigned a pre-defined threshold default, and each subsequent block would use the statistic calculated from the previous block.

This approach has been used for the Magellan mission to venus [KwJo89], and was found to make very little SQNR difference, but results in substantial memory reductions. The question still remaining, is how to calculate the standard deviation in an efficient way.

In [KwJo89], the average magnitude was used instead of the variance to determine the quantization level for the SAR data blocks. As discussed in Section 6.1.1, the standard deviation σ is related to the average magnitude $|\bar{X}|$ by

$$\sigma \approx 1.25|\bar{X}| \tag{6.17}$$

and the threshold b for the 2-bit quantizer is then

$$b \approx 0.9816(1.25|\bar{X}|) \tag{6.18}$$

The calculation of Eq. 6.18 is still a floating-point operation, and is not desirable for on-board computation. An alternative is to have all possible thresholds pre-computed and stored in a look up table where each value in the look up is equal to

$$LUT[i] = \lfloor 0.9816(1.25i) \rfloor \tag{6.19}$$

where i is the address in the look up table and corresponds to $\bar{X}$.

The calculation of $|\bar{X}|$ is the sum of the magnitude of each sample for a data block divided by the number of samples in the block. To reduce the complexity in the calculation of $|\bar{X}|$, a sum of the magnitude of each sample could be calculated with only the most significant N bits of the sum used as the address thereby diving the sum by the block size. This is the approach used in [KwJo89], but is only accurate for block sizes that are powers of two. To accommodate any sized block size, the address of the look up table could be the sum of the magnitude and the look-up table would be programmed with

$$LUT[i] = \left\lfloor \frac{0.9816(1.25i)S}{B} \right\rfloor \tag{6.20}$$

where S is the sample magnitude accumulator size in bits minus the number of look-up table address lines and B is the number of samples per block. By programming the look up table in this way, a free running accumulator can be used to progressively calculate the magnitude sum. When the appropriate number of samples have been summed, corresponding to the block size, the accumulator value would then be used to load a new threshold corresponding to Eq. 6.20. The purpose of S in Eq. 6.20 is to allow the use of any sized accumulator and any sized look-up table, two parameters which are implementation dependant.

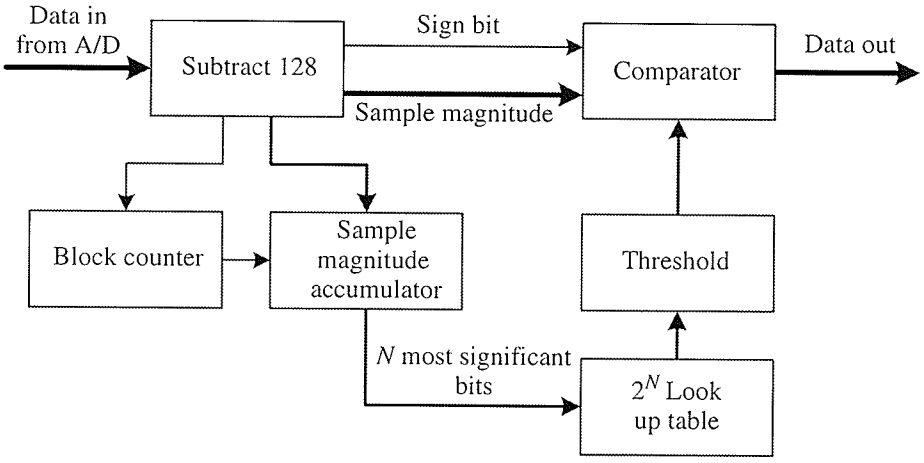


Fig. 6.12. Block diagram of BAQ hardware implementation.

The hardware implementation of the BAQ algorithm can then be described in five steps, as shown in Table 6.2.

Table 6.2 BAQ algorithm for hardware implementation.

Step	Description
1	Read range data sample, 1 at a time. Increment a counter for each read.
2	Subtract 128 from each sample.
3	Add the magnitude of the sample to an accumulator.
4	Compare the magnitude of each sample to the current threshold. Represent the data as: 00 if magnitude is < threshold and sign is 0 01 if magnitude is > threshold and sign is 0 10 if magnitude is < threshold and sign is 1 11 if magnitude is > threshold and sign is 1
5	If the counter has reached the block size, clear the counter and load a new threshold from the look-up table using the accumulator as the address.

6.3.2 Implementation Using the Ballynuey 2 PCI Card

The algorithm of the BAQ hardware implementation discussed in the Section 6.3.1 is easily implementable as either a sequential or a pipelined implementation. Both implementations are possible, however, the pipelined implementation has the added benefit of increased throughput due to the inherent parallelism at the expense of increased interfacing overhead. Additionally, the pipelined approach has the benefit of being able to run slower than a comparable sequential implementation. Consider the criteria of processing data at 64.5 MS/s. For a pipelined implementation, the pipeline must read in the data at 64.5 MS/s, thereby operating at 64.5 MHz and independent of the number of pipeline stages. In the sequential implementation, the entire algorithm must be run on a piece of data before the next data arrives. This then implies an

operating frequency of 64.5 MHz times the number of FSM stages in the implementation. For a given implementation, the pipeline architecture presents greater room for frequency expansion.

Pipelined implementations of algorithms are very amenable to FPGA implementations due to the ability to place parallelism where needed, and therefore the pipelined approach will be used to implement the hardware BAQ model. The pipelined implementation of the BAQ algorithm presented in Table 6.2 can be broken down into three pipeline stages as shown in Table 6.3.

Table 6.3 Pipelined BAQ algorithm.

Pipeline stage	Description
1	Read range data sample, 1 at a time and subtract 128 from each sample. Increment a counter for each read.
2	Add the magnitude of the sample to an accumulator. If the counter has reached the block size, clear the counter and load a new threshold from the look up table for the address corresponding to the accumulator.
3	Compare the magnitude of each sample to the threshold. Represent the data as: 00 if magnitude is < threshold and sign is 0 01 if magnitude is > threshold and sign is 0 10 if magnitude is < threshold and sign is 1 11if magnitude is > threshold and sign is 1

Each of the pipeline stages communicates with the next stage via an 8-bit data bus and a 1-bit enable signal. At reset all pipeline stages are disabled and will not process any data. The pipeline is then filled from stage 1 to stage 3, where at each stage data is

processed and driven on the data bus, and if valid data is driven on the bus the enable line is also activated. This allows the pipeline to fill up and empty out as data sequences pass though. Clear and load signals are also added from the block counter to the magnitude accumulator and the threshold register respectively. When the block counter is equal to the block size, the block counter will clear the magnitude accumulator and load the current output of the look up table into the threshold register. As the magnitude accumulator is connected to the address bus of the look up table, there is always data at the input to the threshold register, but the input only registered when the counter drives the load signal. In the event that the accumulator is greater than the N address lines of the look-up table, only the N most significant bits of the accumulator will be used to drive the look-up table address bus.

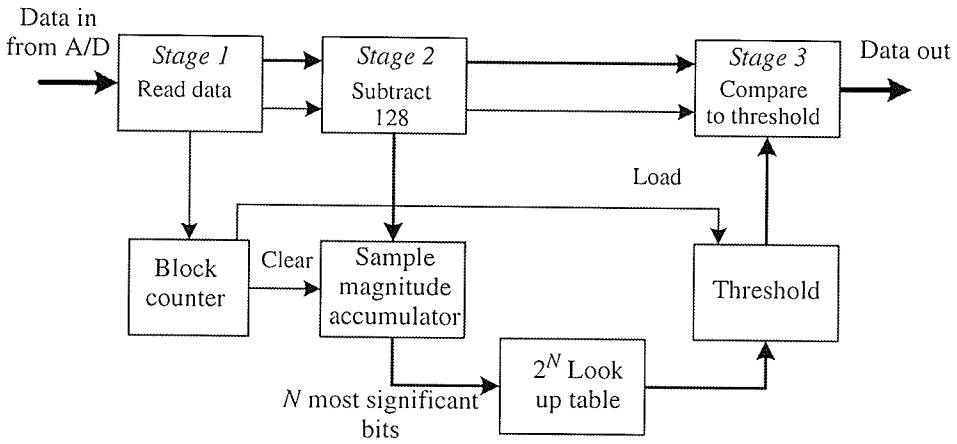


Fig. 6.13. Pipelined implementation of BAQ algorithm.

To map the pipelined BAQ algorithm in the Ballynuey 2 target platform, the algorithm must be implemented as a user core to interface with the PCI core discussed in Section 6.2.1. As a result, an additional pipeline stage will be necessary to write the

BAQ codewords from the user core to PCI core FIFO. The ZBT RAM will be used to store the threshold look up table. Because the RAM is late-late-read RAM, two additional stages are needed as the appropriate threshold value will appear two clock cycles after the address is registered. As a result, the block counter will activate the accumulator clear signal when the block counter reaches the block size, and then activate the load signal two clock cycles later. An additional requirement, due to the interface with the PCI core, is the ability to stall the pipeline if the user core to PCI core FIFO becomes full. Each pipeline stage is fed with the FIFO full signal, which will instruct each stage to stall and not perform their task until the signal is deactivated. A block diagram of the pipelined implementation of the BAQ algorithm targeted for the Ballynuey 2 PCI card is shown in Fig. 6.14.

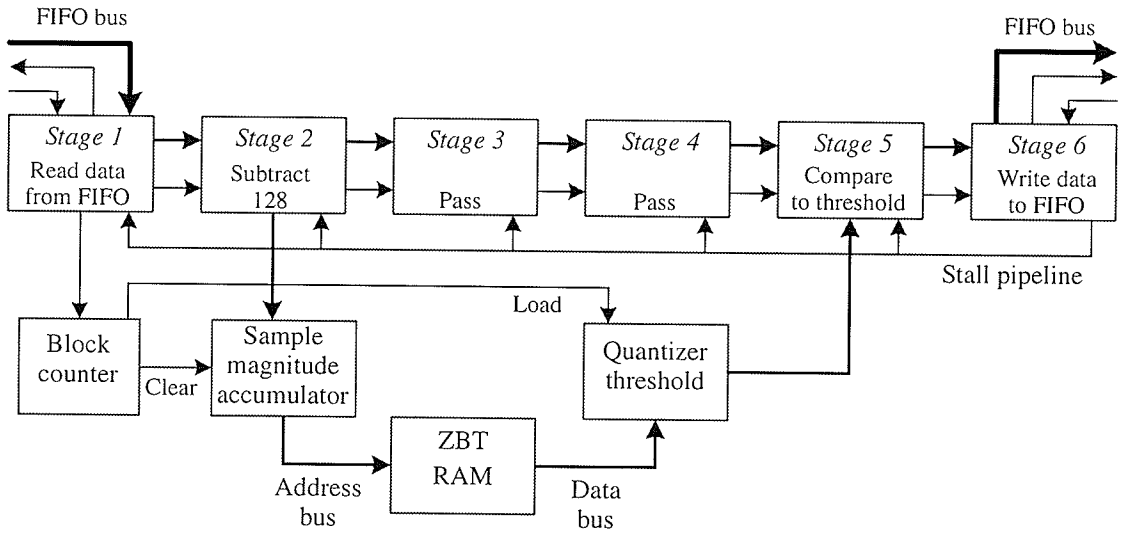


Fig. 6.14. Pipelined implementation of BAQ on the Ballynuey 2 PCI card.

The VHDL code BAQ core developed for this thesis is given in Appendix D. From the code, there are two generic parameters that are passed to the core, CounterWidth

and BlockSize. These two parameters change the size of the magnitude accumulator, which subsequently changes what signals drive the 17-bit ZBT address bus and the value where the block counter will generate the clear signal. These two settings exist because for different raw SAR data test sets there may be different block sizes. As a result of changing the block size the maximum value of the counter, defined as 127 times the block size, will dictate the counter size necessary. With the counter size defined, only the most significant 17 bits are driven to the ZBT address bus, where the ZBT RAM is programmed according to Eq. 6.21.

6.3.3 FPGA Implementation of MRA

Using the polyphase decomposition of a filter described in Section 6.1, the MRA analysis can be efficiently performed by the filtering of even and odd samples, thus removing the need for unused arithmetic and downsampling. The basic MRA analysis algorithm block diagram shown in Fig. 6.4 can be further simplified using the MRA filter identity

$$\tilde{g}[n] = (-1)^n \tilde{h}[-n] \tag{6.21}$$

resulting in the figure shown in Fig. 6.15, for an $\tilde{h}$ filter of length 4. The only parameter left to define are the coefficients of the $\tilde{h}$ filter.

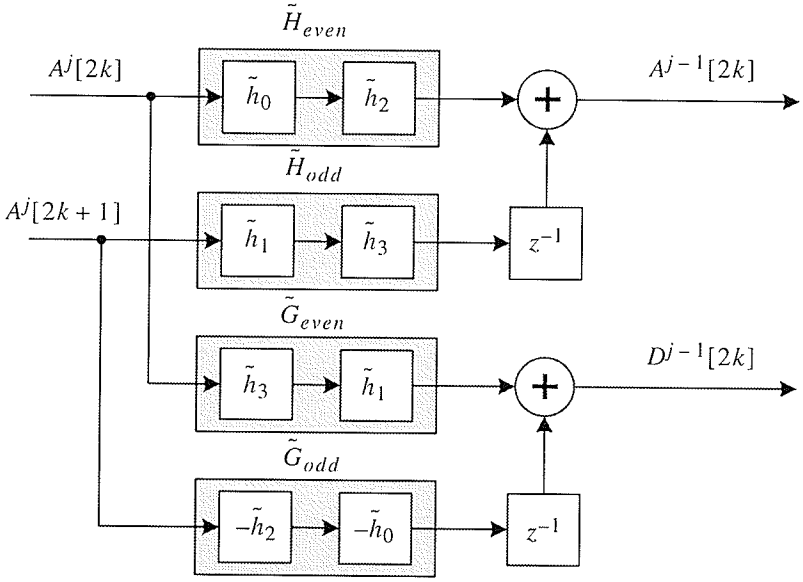


Fig. 6.15. Polyphase implementation of a MRA filter of length 4.

Defining the filter coefficients is not as evident as it may appear since the filter coefficients for the chosen Daubechies-2 wavelet are all rational numbers and, as previously stated, the desired CCE is an integer only computing machine. To provide a solution to this problem, recall that the approximation coefficients are given by

$$a_k^1 = \tilde{h}_0 a_{2k}^0 + \tilde{h}_1 a_{2k+1}^0 + \tilde{h}_2 a_{2k+2}^0 + \tilde{h}_3 a_{2k+3}^0 \tag{6.22}$$

Consider then, that if the $\tilde{h}$ coefficients are lifted [Meye01] by a factor for L so that $\tilde{h} = \frac{h'}{L}$ then

$$a_k^1 = \frac{(h_0' a_{2k}^0 + h_1' a_{2k+1}^0 + h_2' a_{2k+2}^0 + h_3' a_{2k+3}^0)}{L} \tag{6.23}$$

By choosing an appropriate lifting factor, the new filter coefficients h' will all be integer. Once the polyphase convolution is complete, the resulting A^1 and D^1 signals can then be obtained by a simple division by L.

Implementing these filter coefficients effectively involves determining the factor L which will give the best representation. To facilitate the removal of the factor, the logical factor would be a power of 2. The use of a signed 9-bit number essentially creates a positive or negative fraction over 256 for the coefficients and provides for adequate accuracy to the originals. The lifted coefficients for the db-2 wavelet are shown in Table 6.4.

Table 6.4 Daubechies-2 filter coefficients and lifted coefficients

Coefficient index	$\tilde{h}$ coefficients	Lifted $\tilde{h}$ coefficients	Value of lifted coefficient
0	0.4829629131	124	0.484375
1	0.8365163037	214	0.8359375
2	0.2241438680	57	0.22265625
3	-0.1294095225	-33	-0.12890625

With all aspects of the integer only algorithm defined, the last step is a decision on the architecture. As with BAQ, the basic choice is between a sequential or pipelined architecture. The polyphase MRA is inherently pipelined, with the filtering of the incoming data, and when added to the previous decision about a pipelined BAQ, the logical choice is also a pipelined MRA. Using the polyphase MRA, the resulting

transform will produce one approximation or detail coefficient at every clock cycle, since no downsampling is necessary. The output of the polyphase MRA transformer can then be directly input into two BAQ modules, one to quantize the A^1 signal and one to quantize the D^1 signal.

With all the architecture parameters of the MRA transformer fixed, the hardware implementation can be discussed in more detail. As previously mentioned, the FIR operation of the MRA is inherently pipelined, and as such it makes sense to implement it that way. There is still the issue of the delay necessary for the odd part of the filter. If this delay is placed at the beginning of the pipeline, then once the pipeline is full, the result from each of the four multipliers for approximation and detail signal can be summed together to produce the transformed coefficient. This eliminates the need to register individual multiplication and ensure that they are delayed by the proper number of clock cycles.

This design choice then simply requires an additional sorter at the transformer to place the odd samples into one pipeline and the even samples into the other. A block diagram for the hardware implementation of the db-2 MRA transformer is shown in Fig. 6.16.

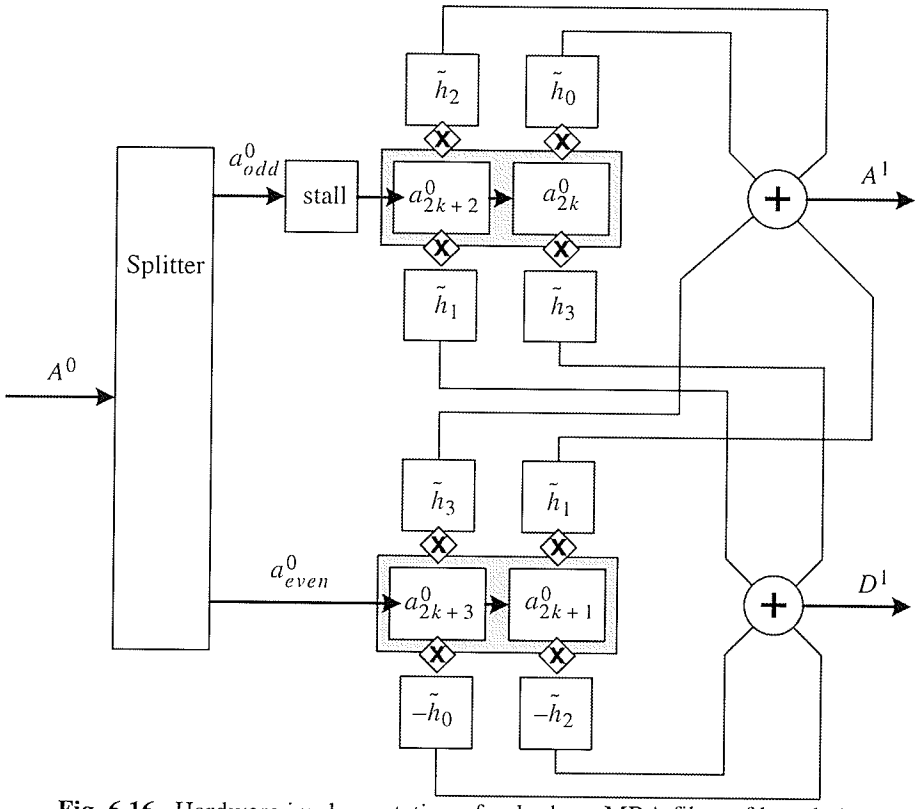


Fig. 6.16. Hardware implementation of polyphase MRA filter of length 4.

The system consists of first splitting the block of range data into even and odd samples. These samples are then fed into the multiplier pipeline where they are multiplied with the 9-bit filter coefficients. Once the multiplier pipeline is full, one approximation or detail coefficient will appear at every clock cycle. That is to say that with an input data rate of R, each of the even and odd data streams will be $R/2$ and so too will be the approximation and detail signals. Since the coefficients are 9-bits the output of the multiplication will be 17-bits. The BAQ must then be configured to quantize 17-bit data. The BAQ used in the wavelet-BAQ is identical to the BAQ previously discussed except that the pipeline stage to subtract 128 from each sample has been removed. VHDL code for the wave-BAQ implementation is listed in

Appendix D. A complete block diagram of the 1-D wavelet-BAQ implementation is show in Fig. 6.17.

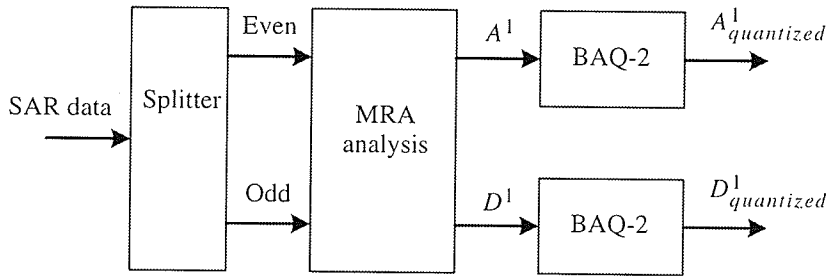


Fig. 6.17. VHDL Implementation of wavelet-BAQ.

As with the BAQ, the wavelet-BAQ has several generic parameters that allow the customization of the implementation. These parameters are the multiplier coefficients, the data with the multiplier output, the size of the BAQ accumulator and the block size. As this implementation uses only the db-2 wavelet, the multiplier coefficients used are those from Table 6.4. The multiplier output is also fixed to 17-bits as this is the required number of bits for the multiplication of an 8-bit and 9-bit number. The choice of 17-bits is also enough to store the sum of the four multiplier results, but would have to be expanded if more than four multiplications were summed.

The last parameters to specify are the BAQ accumulator and the block size. Both of these parameters are chosen to reflect the data set used, as the block size for the data sets are different. As with the BAQ, the accumulator is chosen to be large enough to hold block size number of 17-bit numbers, the top 16-bits of the accumulator are then used as the address lines for the ZBT-RAM LUT. Because the 1-D wavelet-BAQ implementation uses 2 BAQs to quantize the MRA coefficients, two look-up tables are

necessary. This is easily accommodated as there are two banks of ZBT RAM on the Ballynuey 2 PCI card. Both banks are programmed with the same look up table corresponding to Eq. 6.20. Table 6.5 lists the block size and accumulator width for each of the test sets used in this thesis.

Table 6.5 Block size and counter width for pre-conditioned test sets.

Test set	Total samples per range line	BAQ block size	Accumulator width (bits)
E1-22089PS	11088	1386	27
E1-25224PS	11232	1404	27
E2-5551PS	11232	1404	27
R1-24576PS	13416	1677	27
R1-24919PS	13416	1677	27

6.4 Summary

With the development of the 1-D wavelet BAQ of Chapter 5, the last step was its implementation in hardware. Ideally, a raw SAR data development kit would be used as the target for the implementation, but unfortunately these are not available. The 1-D wavelet BAQ, and the BAQ alone, were therefore implemented using the Ballynuey 2 PCI carrier card from Nallatech.

The implementation of the algorithms had the criteria of a low complexity integer only arithmetic CCE, with minimal memory requirements. Because of the disallowance of floating-point math, a direct implementation of the MatLab 1-D

wavelet-BAQ was not possible and a re-examination of the 1-D wavelet-BAQ algorithm was necessary.

To implement the BAQ in low complexity hardware, the calculation of the quantizer thresholds was performed using the average magnitude statistic, instead of the standard deviation, and the statistic was calculated on the previous block of data. To further simplify the calculation of the quantizer threshold, all thresholds were programmed into a look-up table addressed using the accumulated signal magnitude of the block. The delayed calculation of the statistic was possible by the observation of slowly changing signal power in both azimuth and range directions.

Likewise, with the implementation of the 1-D wavelet BAQ there was the need to optimize some procedures for the chosen architecture. A significant reduction in computation, and a subsequent increase in performance was possible using the polyphase decomposition of the MRA analysis algorithm. This technique allows the direct calculation of the MRA coefficients from a convolution without the need for a downsampling. An additional issue with the MRA implementation was the use of integer filter coefficients. This issue was resolved by lifting the coefficients by 256, essentially making them signed fractions over 256. The numerator of the lifted coefficients was then used in the calculation, thereby making it an integer only operation.

CHAPTER 7

RESULTS OF THE 1-D WAVELET-BAQ TECHNIQUE

In the previous chapter the 1-D wavelet BAQ algorithm was implemented as a low complexity CCE in Xilinx FPGA hardware. This implementation is the major work of this thesis, and it is the focus of this chapter to evaluate the implementation. This chapter will compare the 1-D wavelet BAQ CCE against the BAQ CCE using metrics in both the signal domain and processed image domain.

This chapter begins with a review of current metrics for raw SAR data compression in an attempt to determine the best subset of metrics to evaluate the 1-D wavelet BAQ. Using the chosen metrics, the results of the BAQ and 1-D wavelet BAQ implementations run on the five pre-conditioned raw SAR data sets will be analyzed. Using the metrics, an evaluation of the 1-D wavelet BAQ will be made to determine its relative merits and shortcomings. The chapter concludes with a brief discussion on some of the operating parameters of the FPGA implementations.

7.1 Metrics for Raw SAR Data Compression

When evaluating a raw SAR data compression technique, both the signal domain and the processed image domain must be considered as the statistics in the signal and image domains are quite different and metrics used in one domain are not always applicable in the other. The basic test model used in the literature for evaluating a new

raw SAR data compression technique is to measure the effect of the quantization in the signal domain and then if the signal domain results are promising, process both the original and reconstructed data set to compare them in the image domain. A block diagram of the typical raw SAR data compression technique evaluation environment is shown in Fig 7.1.

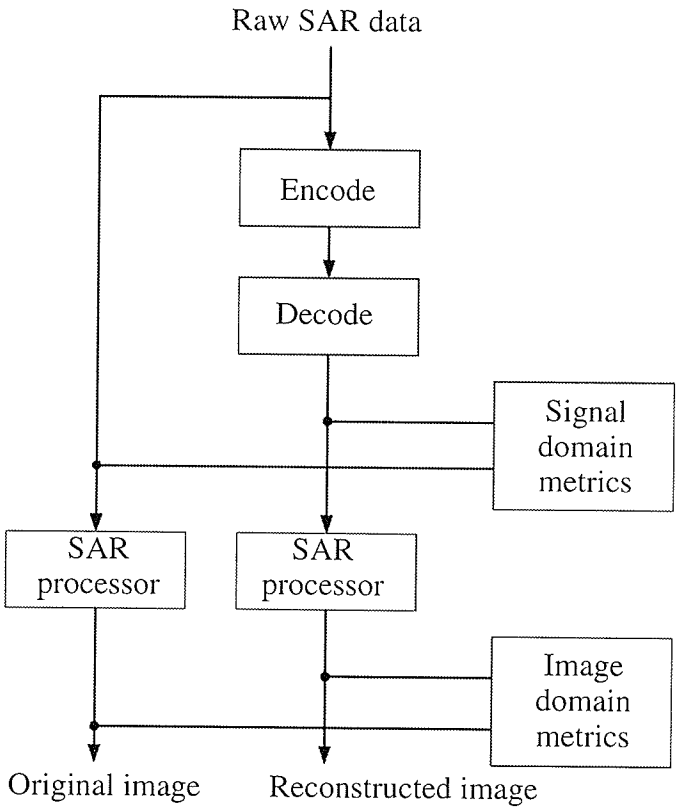


Fig. 7.1. Evaluation environment for SAR data compression techniques (from [DuCu94]).

The standard signal domain metric used by almost all papers discussing raw SAR data compression is the SQNR. There have been several studies discussing only SAR data metrics and their ability to evaluate the effects of quantization [DuCu92], [DuCu94], [McCS98].

Another area where SAR data metrics are used is to evaluate SAR processors by their effects on the resulting SAR images. Several papers have been written in this area [ArVE99], [Harr87], [HOCC88], and these proposed metrics are also applicable in evaluating the changes to the processed SAR images due to quantization of the raw SAR data.

7.1.1 Metrics in the Signal Domain

To evaluate the effect of the quantization of the raw SAR data [DuCu92], [DuCu94] have suggested the following metrics:

(1) Comparison of the data mean, standard deviation, kurtosis and entropy of the original and quantized SAR data. These can be used to highlight any changes in the data statistics.

(2) The signal to quantization noise ratio (SQNR) and peak signal to quantization noise ratio (PSQNR). These global metrics give relative measures of the error in an absolute sense and are useful in comparing results of encoding between different data sets with different statistics.

(3) Comparison of the histograms of the original and quantized data sets. This will highlight any changes in the probability distribution due to the quantization. The histogram of the error is also useful as it will highlight the probability distribution of the error due to quantization.

(4) An examination of the phase error as a technique which produces severe phase distortion may be unusable. The phase information in the raw SAR data is used by the SAR processor to focus the processed SAR image and in interferometry [McCS98].

7.1.2 Metrics in the SAR Image Domain

To evaluate the quality of the processed SAR image, [ArVE99], [Harr87], and [DuCu94] have proposed the following SAR image domain metrics:

(1) Comparison of the data mean, standard deviation, kurtosis and entropy of the SAR images. These statistics can be used to highlight any changes in the data statistics.

(2) The SQNR and PSNR of the SAR images. These global metrics are accepted by [DuCu94], but strongly rejected by both [ArVE99] and [Harr87] claiming the quantization of the raw SAR data degrades the SAR image produced from the original data by non-linear distortion, linear distortion, and additive noise. The linear noise then implies correlation between the SAR images from the original and quantized data, which violates the assumptions made by SQNR and PSNR.

(3) Comparison of the SAR image histograms from the original and quantized data sets. Examining the histograms will highlight any changes in the probability

distribution due to the distortion in the signal domain. The histogram of the error is also useful as it will hightlight the probability distribution of the error due to quantization.

(4) Comparison of the 1-D and 2-D image spectra of the SAR images from the original and quantized data sets to highlight any discrete frequency artifacts, differences in bandwidth and differences in frequency content. The 1-D and 2-D spectra of the error image can also be examined to show whether the error is greater for high or low frequency image components.

(5) The normalized integrated side lobe ratio (NISLR) is a normalized version of the integrated side lobe ratio (ISLR) which is the ratio of the total energy returned from a point to the energy contained outside the mainlobe. The normalizing factor addresses the problem of the ISLR in comparison of quantized data which decreases the energy on the mainlobe and sidelobes.

It was the conclusion of [DuCu94] that the single most important metrics are the image domain SQNR and the signal domain SQNR as they give a measure of the severity of the unstructured error due to quantization. The SQNR is not a perfect solution as noted by [ArVE99] and [Harr87], since it is a global measure and gives no information about the spacial location of the error or indication to the extent to which the error effects the end use of the image. Therefore, in order to gain additional information about the characteristics of the error, [DuCu94] suggest using: (i) the data

statistics of the raw data and image data, (ii) comparison of the histograms, (iii) error images, and (iv) image spectra.

These conclusion are not rejected by [ArVE99], [Harr87], [HOCC88] except for the use of the SQNR on image data. These researchers assert that while the SNQR in general is higher for visually superior images, a given value of SQNR does not indicate a corresponding level of image quality and may be misleading. They believe that the NISLR is a metric that should be included in any measure of SAR image data.

7.2 Design of Experiments

7.2.1 Metrics

Although an exhaustive analysis using all possible metrics would be ideal, this thesis will limit the metrics to only a few. The metrics chosen reflect the most common metrics used in the literature, and focus mainly on the performance in the signal domain.

For this thesis, the signal domain SQNR quantization gain, entropy, phase error and histogram analysis will be used as metrics along with the image domain SQNR and PSNR and histogram analysis. The histogram analysis will investigate the original and reconstructed histograms by presenting figures along with the variance of the error.

7.2.2 BAQ

To implement the BAQ CCE discussed in Chapter 6 using the Ballynuey 2 PCI card, the BAQ core was implemented as a user core and combined with the PCI core. The complete VHDL description was then synthesized using Leonardo Spectrum 2002b21 and implemented using the Xilinx ISE 4.2iSP2 place-and-route tools. The user FPGA was then programmed using the resulting FPGA bitstream using the Nallatech low-level drivers.

With the Ballynuey 2 programmed with the implementation of BAQ, each of the five pre-conditioned raw SAR data test sets was in-turn written to the PCI FPGA with the resulting codewords read. Since each codeword represents the 2-bit BAQ codeword and the 18-bit accumulator value corresponding to the threshold, a reconstructed data file could be created.

With the reconstructed data files created, the metrics described in Section 7.2.1 can be calculated.

7.2.3 Wavelet-BAQ

As with the BAQ, the 1-D wavelet-BAQ core must be first combined with the PCI core for implementation on the Ballynuey 2 PCI card. The toplevel design is then synthesized using Leonardo Spectrum and implemented using the Xilinx place-and-route tools. The resulting bitstream is then programmed to the user FPGA of

the Ballynuey 2 using the Nallatech low level drivers. Each of the five test sets are then in-turn written to the PCI FPGA and the resulting quantized MRA coefficients codewords are read. The data file can then be reconstructed by first reconstructing the MRA coefficients, then applying the MRA synthesis algorithm to reconstruct the signal domain data. Both of these steps were programmed in C and run on the resulting data file read from the output of the PCI FPGA. With the data sets reconstructed after the 1-D wavelet-BAQ, the metrics described in Section 7.2.1 can be run.

7.3 Experimental Results

7.3.1 BAQ

7.3.1.1 Signal Domain

Signal domain results of the five pre-conditioned test sets quantized using the FPGA implementation of the BAQ-2 algorithm are shown in Table 7.1 These results are compared with the BAQ-2 algorithm programmed in MatLab. The purpose of presenting these results is simply to verify that the FPGA-BAQ performs as expected, that way it can be used as an appropriate benchmark for the performance of the 1-D wavelet-BAQ FPGA implementation.

As can be seen from Table 7.1, the SQNR of the FPGA BAQ-2 is not as high as the MatLab BAQ-2, performing, on average, 0.2 dB lower than the MatLab BAQ. This is to be expected as the FPGA BAQ-2 does not calculate the standard deviation of each

block before quantizing it. The quantization gain is slightly higher for the FPGA BAQ, on average 0.13 V/V higher than the MatLab BAQ, again an artifact of the late estimate of the block variance. Although the FPGA BAQ gain is slightly higher, it is not a significant gain.

Table 7.1 SQNR of the FPGA BAQ for the E1-22089PS, E1-25224PS, E2-5551PS, R1-24919PS and R1-24576PS test sets compared the MatLab BAQ.

Test set	SQNR in dB		Quantization gain $\frac{\sigma_q^2}{\sigma_{raw}^2}$	
	FPGA BAQ-2	MatLab BAQ-2	FPGA BAQ-2	MatLab BAQ-2
Mean	9.35	9.52	0.902	0.889
E1-22089PS	9.36	9.60	0.912	0.899
E1-25224PS	9.68	9.80	0.931	0.908
E2-5551PS	9.24	9.36	0.913	0.888
R1-24919PS	9.25	9.41	0.880	0.877
R1-24576PS	9.20	9.45	0.875	0.874

An analysis of the FPGA BAQ histograms also show the expected results, as shown in Fig. 7.2, and complete histograms in Appendix F. The histogram in Fig. 7.2 shows the four reconstruction levels of the quantizer and its variation as the signal statistics change. Comparing this to the histogram of the same data set but quantized using the MatLab BAQ-2, shown in Fig. 7.3 with complete histograms shown in Appendix F, shows virtually the same shape. Numerical results from the calculation of the entropy and phase error for the FPGA BAQ-2 and MatLab BAQ-2 is shown in Table 7.2.

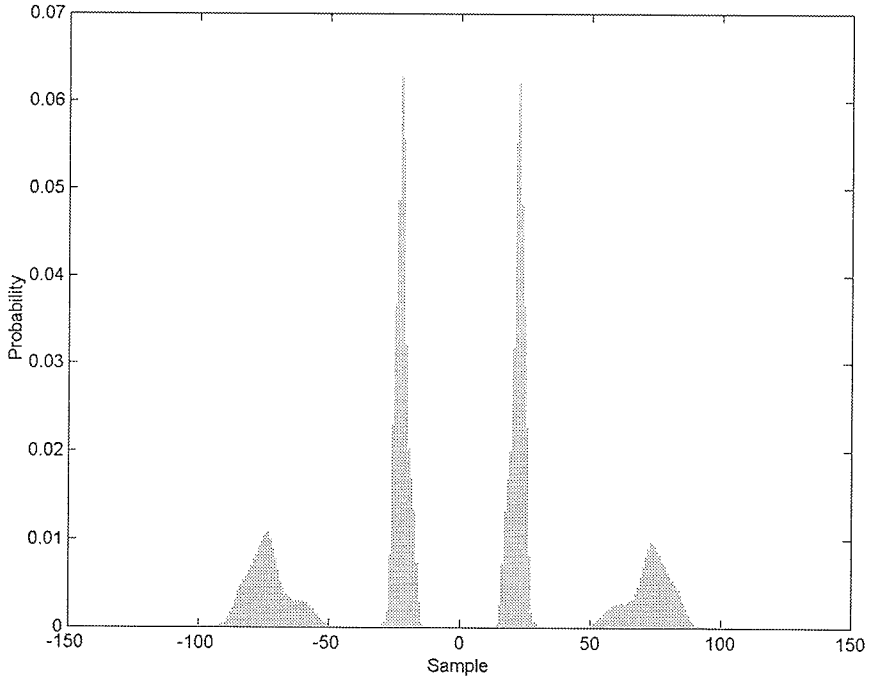


Fig. 7.2. Histogram of the E1-25224PS test set quantized with FPGA BAQ-2.

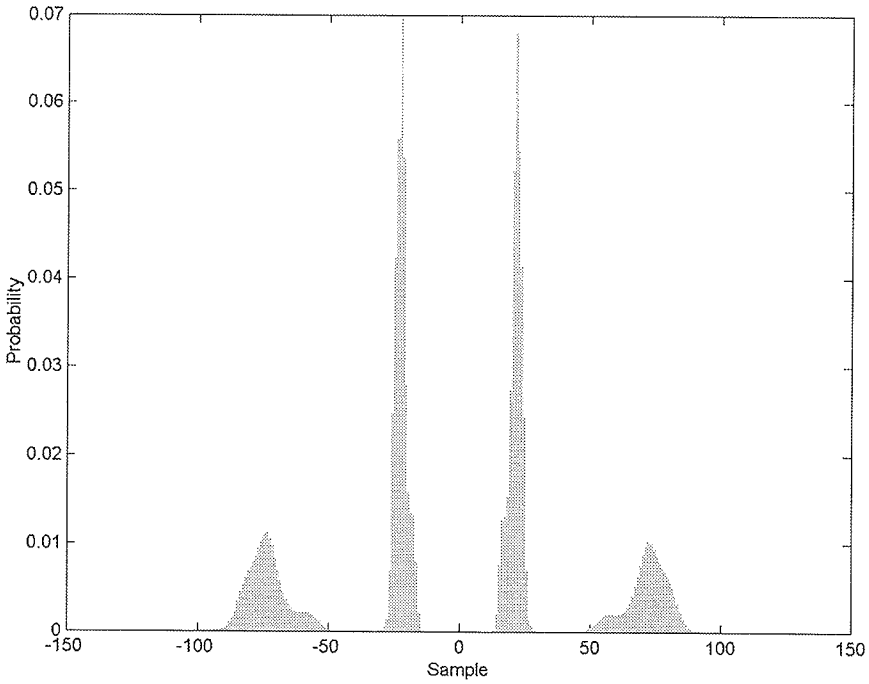


Fig. 7.3. Histogram of the E1-25224PS test set quantized with MatLab BAQ-2.

Table 7.2 Entropy, and phase error of the FPGA BAQ for the E1-22089PS, E1-25224PS, E2-5551PS, R1-24919PS and R1-24576PS test sets compared the MatLab BAQ.

Test set	Entropy in bits		Phase error variance	
	FPGA BAQ-2	MatLab BAQ-2	MatLab BAQ-2	FPGA BAQ-2
Mean	5.49	5.58	0.101	3.567
E1-22089PS	5.90	5.92	0.100	3.534
E1-25224PS	5.70	5.81	0.099	3.555
E2-5551PS	5.32	5.36	0.101	3.519
R1-24919PS	5.26	5.37	0.103	3.613
R1-24576PS	5.26	5.42	0.104	3.615

The entropy of the FPGA and MatLab implementations of BAQ are very similar, with an average difference of only 0.09 bits, making it a fairly insignificant difference. The phase error variance is the first numerical result that very strongly emphasizes the difference between the MatLab and FPGA implementations. These numbers are even more pronounced when examining the phase error histograms, as shown in Fig. 7.4 and Fig. 7.5 with complete histograms of all test sets shown in Appendix F. The magnitude of the phase error is directly related to the lack of precision in the calculation of the block standard deviation and subsequently the quantizer threshold.

From the data in Table 7.1, Table 7.2, and the histogram figures comparing the FPGA BAQ-2 to the MatLab BAQ-2, it can be concluded that the FPGA BAQ-2 produces data consistent with the MatLab BAQ-2 and can be used as a benchmark representing the performance of a hardware BAQ. This conclusion can be made as all data presented comparing the two implementations produce near identical results.

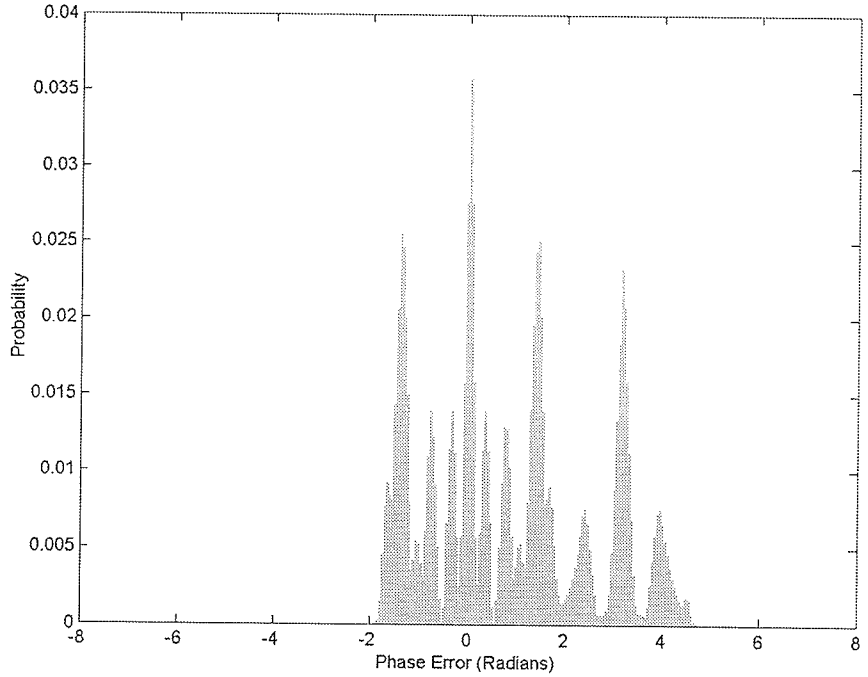


Fig. 7.4. Phase error histogram of the E1-25224PS test set quantized with FPGA BAQ-2.

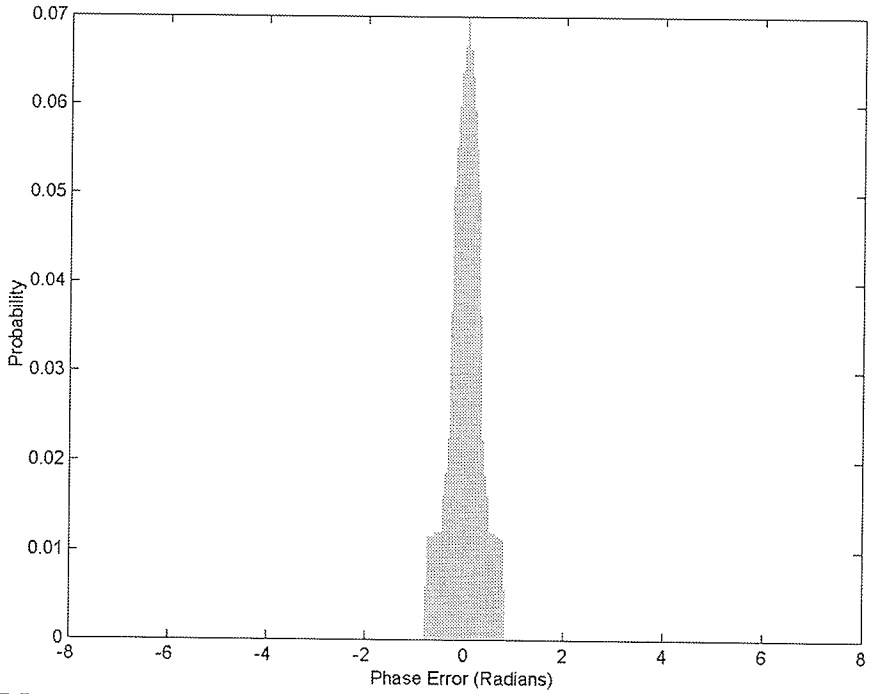


Fig. 7.5. Phase error histogram of the E1-25224PS test set quantized with MatLab BAQ-2.

7.3.2 Wavelet-BAQ

7.3.2.1 Signal Domain

Using the 1-D wavelet-BAQ developed in Chapter 6, SQNR results and quantization gain for each of the pre-conditioned test sets are shown in Table 7.3, with entropy and error variance for the test sets is listed in Table 7.4 and Table 7.5 respectively.

Table 7.3 SQNR of the FPGA MRA core for the E1-22089PS, E1-25224PS, E2-5551PS, R1-24919PS and R1-24576PS test sets compared to BAQ and the Shannon Bound.

Test set	SQNR in dB		Quantization gain $\frac{\sigma_q^2}{\sigma_{raw}^2}$	
	FPGA BAQ-2	FPGA wavelet-BAQ-2	FPGA BAQ-2	FPGA wavelet-BAQ-2
Mean	9.35	8.95	0.902	0.860
E1-22089PS	9.36	8.74	0.912	0.867
E1-25224PS	9.68	9.32	0.931	0.888
E2-5551PS	9.24	9.07	0.913	0.869
R1-24919PS	9.25	8.81	0.880	0.839
R1-24576PS	9.20	8.78	0.875	0.836

In analysing the results in Table 7.3 it immediately appears that the 1-D wavelet-BAQ performs worse than BAQ as indicated by its consistently lower numerical results. The SQNR of the wavelet-BAQ is on average 0.4 dB lower than the BAQ, as is the quantization gain with is on average 0.42 lower. These numbers are not entirely

unsuprising considering that the MatLab implementation of the 1-D wavelet-BAQ discussed in Chapter 5 did not find any wavelet which had superior SQNR than BAQ.

Table 7.4 Entropy of the FPGA BAQ for the E1-22089PS, E1-25224PS, E2-5551PS, R1-24919PS and R1-24576PS test sets compared the MatLab BAQ.

Test set	Original	FPGA BAQ-2		FPGA wavelet BAQ-2	
	Entropy	Entropy	Δ Entropy	Entropy	Δ Entropy
Mean	7.41	5.49	1.92	7.28	0.13
E1-22089PS	7.47	5.90	1.57	7.35	0.12
E1-25224PS	7.62	5.70	1.92	7.53	0.09
E2-5551PS	7.11	5.32	1.79	6.99	0.12
R1-24919PS	7.43	5.26	2.17	7.28	0.15
R1-24576PS	7.42	5.26	2.16	7.27	0.15

The entropy results present some interesting results suggesting higher information content in the wavelet-BAQ data than the BAQ data. These numbers are consistent with the histograms of the reconstructed wavelet-BAQ data sets as shown in Fig. 7.6, with complete histograms for all data sets shown in Appendix F. When comparing the wavelet-BAQ histograms to the BAQ histograms it is evident that the wavelet-BAQ histogram is much closer to the original histogram than the BAQ histogram. The question is whether this is to due to a change in the noise statistics. This does appear to be the case when examining the entropy of the error signal and the variance of the phase error, shown for both the FPGA BAQ and FPGA wavelet-BAQ in Table 7.5

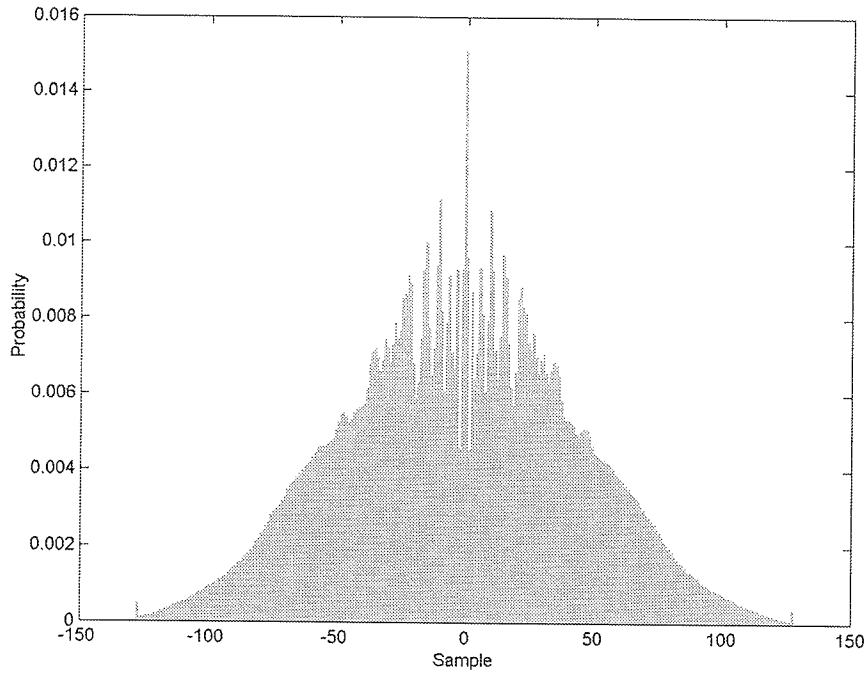


Fig. 7.6. Histogram of the E1-25224PS test set quantized with FPGA wavelet-BAQ-2.

The error data shown in Table 7.5 shows only slightly higher noise energy for the wavelet-BAQ compared to the BAQ for all test sets. This elevation in noise is also visible when examining the error histogram and phase error histogram as shown in Fig. 7.6, with complete histograms for all test sets shown in Appendix F. The wavelet-BAQ error histogram is more of a Gaussian shape compared to the rectangular shape of the BAQ. This can be attributed to the fact that the quantization is performed in the transform domain for the wavelet-BAQ.

Table 7.5 Error entropy and phase error variance of the FPGA BAQ for the E1-22089PS, E1-25224PS, E2-5551PS, R1-24919PS and R1-24576PS test sets compared the MatLab BAQ.

Test set	Entropy of error in bits		Phase error variance	
	FPGA BAQ-2	FPGA wavelet-BAQ-2	FPGA BAQ-2	FPGA wavelet-BAQ-2
Mean	5.49	5.88	3.567	3.880
E1-22089PS	5.90	5.94	3.534	3.887
E1-25224PS	5.70	6.08	3.555	3.843
E2-5551PS	5.32	5.57	3.519	3.860
R1-24919PS	5.26	5.92	3.613	3.901
R1-24576PS	5.26	5.91	3.615	3.910

The phase error histogram is also somewhat different for the wavelet-BAQ compared to the BAQ. The wavelet-BAQ phase error histogram is larger and more spread out, which is reflected by its increased variance.

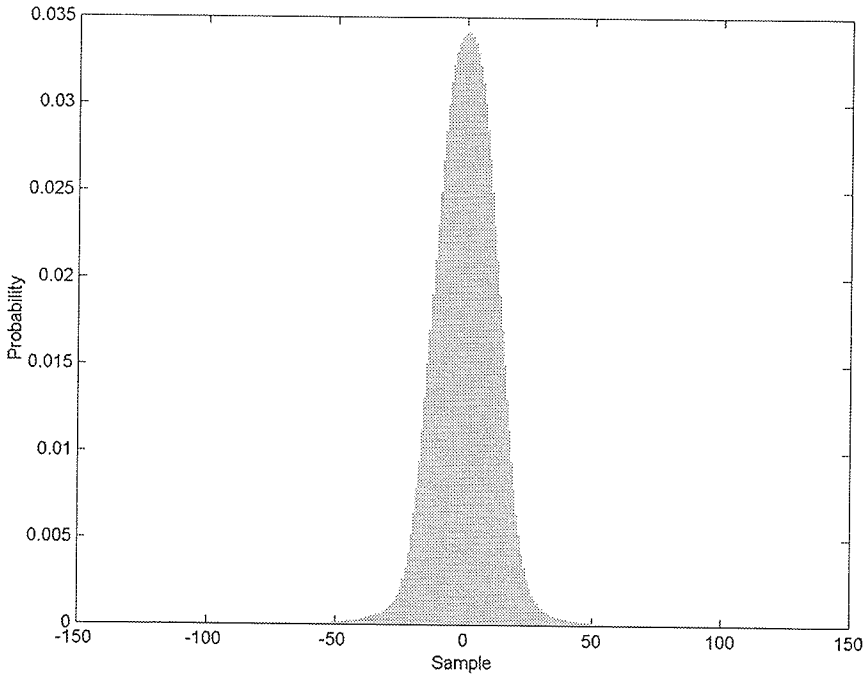


Fig. 7.7. Error histogram of the E1-25224PS test set quantized with FPGA wavelet-BAQ-2.

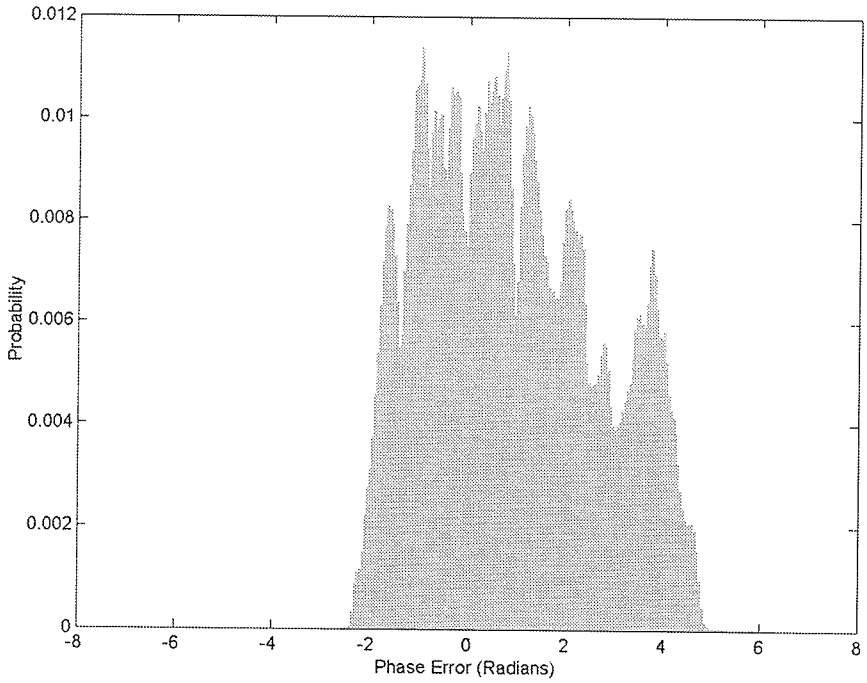


Fig. 7.8. Phase error histogram of the E1-25224PS test set quantized with FPGA wavelet-BAQ-2.

From the results of the wavelet-BAQ compared to the BAQ in the signal domain, the BAQ has consistently performed better than the wavelet-BAQ in all metrics except entropy. It should be noted however that the difference between the BAQ and the wavelet-BAQ is only marginal.

7.3.2.2 Image Domain

To efficiently analyze the effect of the signal domain quantization, the image domain metrics are also necessary. The SQNR and PSQNR image domain metrics for the BAQ and wavelet-BAQ FPGA implementations are shown in Table 7.6 for all test sets.

Table 7.6 Image domain SQNR and PSQNR of the FPGA wavelet-BAQ for the E1-22089PS, E1-25224PS, E2-5551PS, R1-24919PS and R1-24576PS test sets compared the FPGA BAQ.

Test set	SQNR in dB		PSQNR in dB	
	FPGA BAQ-2	FPGA wavelet-BAQ-2	FPGA BAQ-2	FPGA wavelet-BAQ-2
Mean	8.41	7.46	57.11	56.15
E1-22089PS	7.04	4.48	56.13	53.57
E1-25224PS	10.16	9.11	59.06	58.01
E2-5551PS	9.36	8.57	58.61	57.83
R1-24919PS	7.77	7.83	55.16	55.03
R1-24576PS	7.72	7.29	56.59	56.15

The results in Table 7.6 show that the SQNR and PSQNR is lower for the wavelet-BAQ than BAQ for all test sets. These numbers are consistent with the histograms of the error signal as shown in Fig. 7.9 and 7.10, with histograms for all test sets shown in Appendix F. The error histogram shows that the wavelet-BAQ variance is only slightly larger than the BAQ error variance.

The results of the analysis in the image domain is similar to the signal domain, whereby the wavelet-BAQ has consistently performed worse than the BAQ. This statement is justified by the image and signal domain metrics which show no improvement over BAQ. That being said, the deficit is small in all cases and allows for the possibility of improvement if an optimal wavelet for SAR is used.

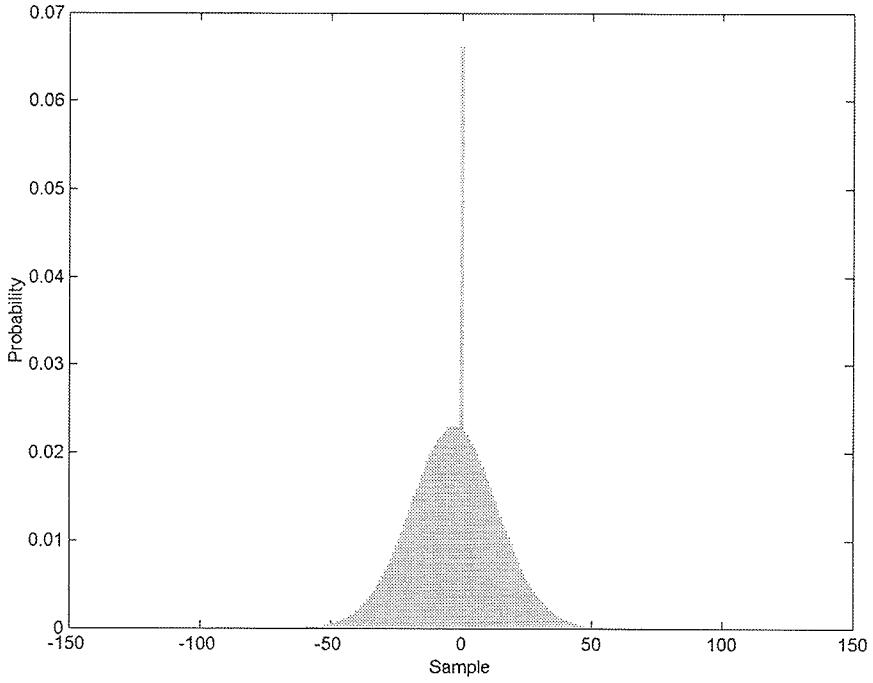


Fig. 7.9. Image error histogram of the E1-25224PS test set quantized with FPGA BAQ-2.

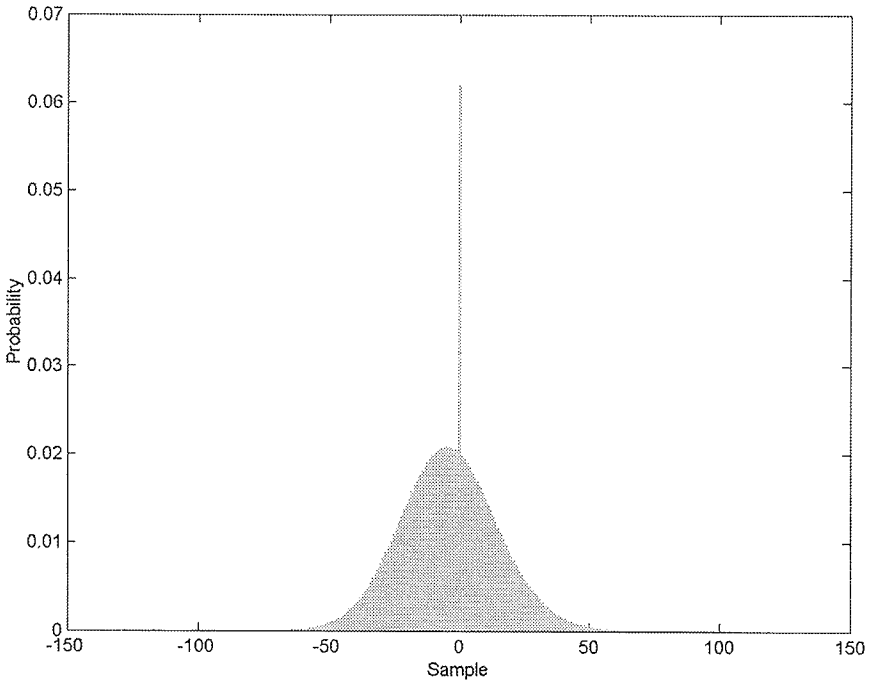


Fig. 7.10. Image error histogram of the E1-25224PS test set quantized with FPGA wavelet-BAQ-2.

7.4 Implementation Results

Both the BAQ and wavelet-BAQ were implemented using the Xilinx place-and-route tools. This section describes some of the parameters about these two implementations.

7.4.1 BAQ

The BAQ core easily fits in the Virtex-600E illustrated by the floorplanner view of the implementation as shown in Fig 7.11.

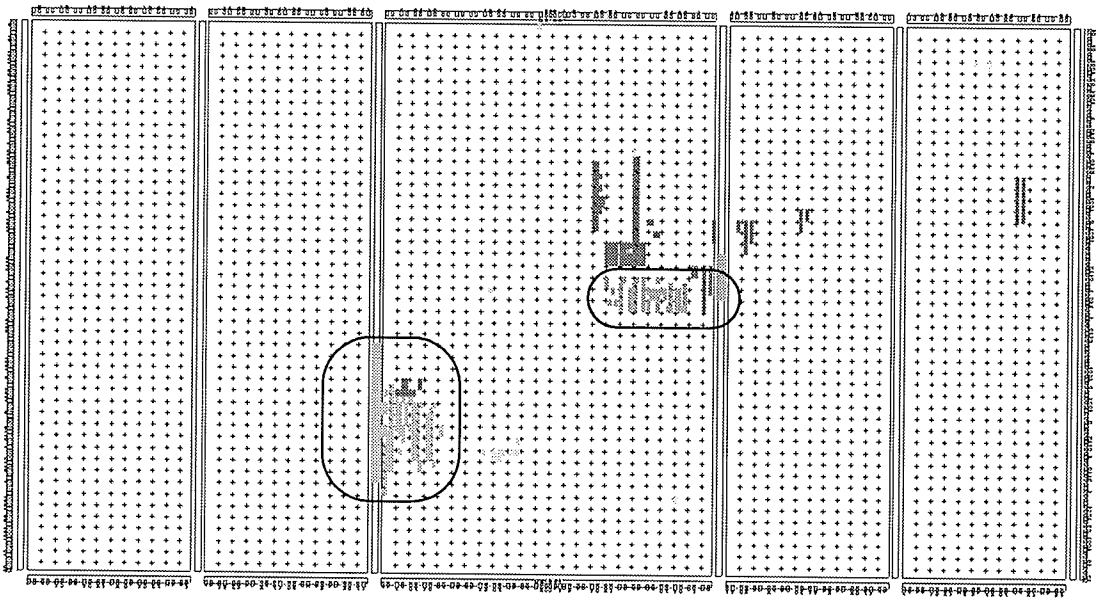


Fig. 7.11. Floorplanner view of BAQ.

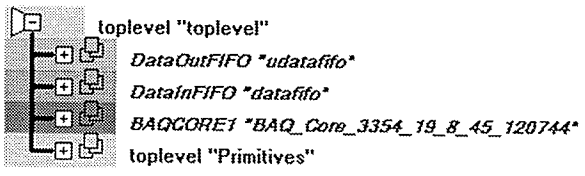


Fig. 7.12. BAQ floorplanner legend.

The floorplanner view shows the logic block allocation to each of the top level instantiated components. The two circled blocks are the data FIFOs and the uncircled block in the upper right quadrant is the BAQ. As can be seen by the large empty spaces, the BAQ implementation takes a very small amount of space.

The synthesis report on the implementation specifies a maximum user core clock frequency of 91.4 MHz, which is close to the actual timing report maximum clock frequency of 59.90 MHz.

7.4.2 Wavelet-BAQ

As with the BAQ implementation, the wavelet-BAQ implementation comes nowhere close to filling the entire FPGA. The implementation does utilize more of the FPGA resources than BAQ, as expected. The floorplanner view of the implementation is shown in Fig. 7.13. The floorplanner shows all logic blocks used by the design and how they are allocated. The two circled blocks represent the two FIFOs used to connect the user core to the PCI core, whereas everything else corresponds to the wavelet-BAQ implementation. By far the largest use of the logic has gone to the multiplier section of the wavelet-BAQ. This is not unexpected since multipliers do occupy more space than adders

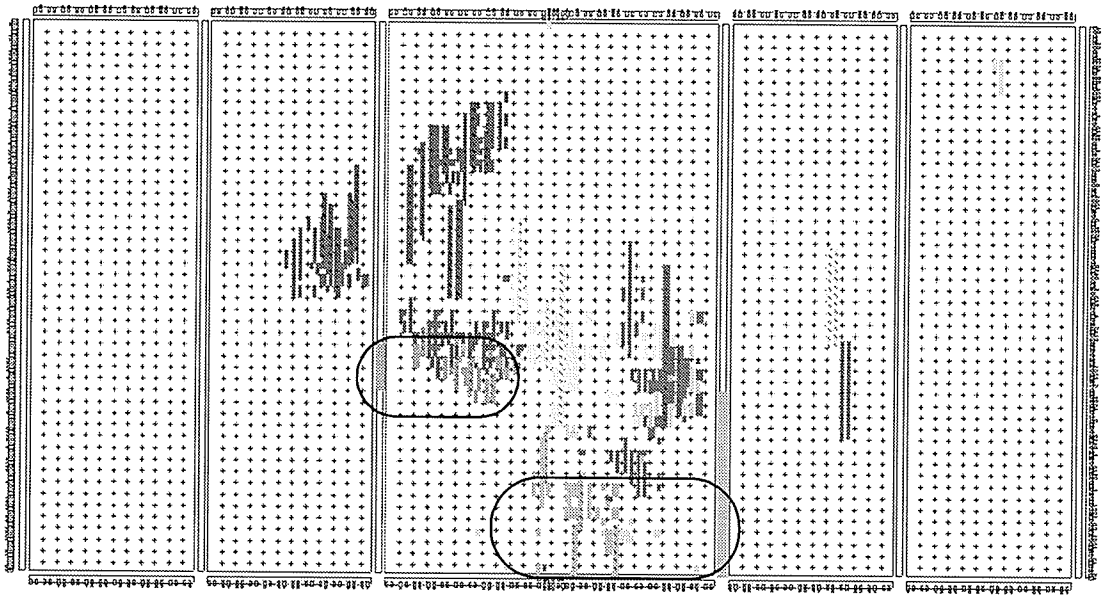


Fig. 7.13. Floorplanner view of wave-BAQ.

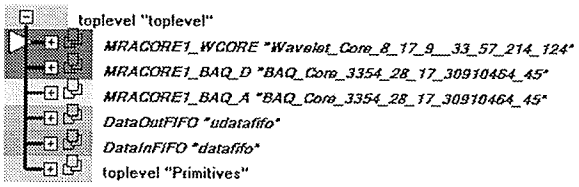


Fig. 7.14. wave-BAQ floorplanner legend.

The maximum clock frequency reported by the place-and route tool for the wavelet-BAQ is 52.47 MHz, slightly higher than the BAQ. This is expected since there are more logic blocks in the wavelet-BAQ.

7.5 Summary

The 1-D wavelet-BAQ FPGA implementation developed in Chapter 6 was compared to the BAQ FPGA implementation using several metrics in both the signal and image domain. Analysis in the two domains is necessary as both domains are used

by researchers. Both the FPGA BAQ and FPGA wavelet-BAQ were programmed into the user FPGA of the Nallatech Ballynuey 2 PCI card for compression of the five pre-conditioned raw SAR data sets. The results of the experiments in both the signal and image domain was that the wavelet-BAQ performed consistently worse than the BAQ. This finding, while disappointing, is not unexpected given the previous results in Chapter 5. Although the BAQ was superior, the difference between the BAQ and wavelet-BAQ was small for all metrics, leaving the possibility for improvement if an optimized wavelet is developed specifically for SAR.

CHAPTER 8

CONCLUSIONS

8.1 Conclusions

Motivated by the development of a high throughput CCE for the compression of raw SAR data with the prospect of surpassing the performance of BAQ, this thesis designed and implemented a 1-D wavelet-BAQ.

The intent was not the development of an improved raw SAR data compression algorithm alone, but also the implementation of the algorithm in low complexity hardware. Throughout the development of the CCE, tests were run on five pre-conditioned raw SAR data sets using all standard wavelets in MatLab, in an attempt to determine the best wavelet basis. It was in doing this extensive testing that it was found that none of the standard wavelet basis produced an SQNR performance greater than BAQ. The different wavelets showed very little variation with all wavelets between Daubechies-1 to Daubechies-44 differing by a maximum of only 0.09 dB for 2 bits/sample quantization. Throughout these tests it was determined that the best performing wavelet was the Daubechies-2 wavelet.

Using the db-2 wavelet, the 1-D wavelet-BAQ algorithm was implemented in Xilinx hardware where the CCE performed on average 0.4 dB worse than BAQ on all test sets. The results of the CCE showed similar results in both the image and signal

domain, performing consistently worse than the BAQ. Although these results were discouraging, the difference between the BAQ and wavelet-BAQ was relatively small presenting the possibility for improvement if a wavelet can be found, or developed, that is better suited to the raw SAR signal.

With the successful hardware implementation of a low complexity raw SAR data compression algorithm with the potential for surpassing BAQ, it is concluded that the objectives of this thesis have been accomplished.

8.2 Contributions

This thesis and the research done towards its completion has provided the following contributions:

1. Extensive testing of 104 standard MRA wavelet basis in MatLab showing that no standard wavelet provides superior SQNR performance than BAQ.

2. The development of a new raw SAR data pre-conditioning technique for reduced dynamic range test sets that produces superior quality test sets than the MDA pre-conditioning technique.

3. The implementation of a 1-D wavelet-BAQ algorithm in low complexity hardware for the compression of raw SAR data. This technique did not provide

an SQNR greater than BAQ, but does have the potential to surpass BAQ if an optimal wavelet can be found.

8.3 Recommendations for Future Work

Based on the work done in this thesis, the following recommendations are suggested for future work in this area:

1. The development of an optimal wavelet for the compression of raw SAR data.

2. Determination of the optimum MRA resolution for a given wavelet.

3. Investigation into additional wavelet techniques such as wavelet packets and the lifting technique.

REFERENCES

[Algr00] T. Algra, "Compression of raw SAR data using entropy-constrained quantization," in *Proceedings of the International Geoscience and Remote Sensing Symposium*, 2000, pp. 2660-2662.

[Arno87] D. V. Arnold, "Vector quantization of synthetic array radar data," M.S. thesis, Brigham Young University, Provo, Utah, USA, 1987.

[ArVE99] G. Arslan, M. Valliappan, and B. L. Evans, "Quality assessment of compression techniques for synthetic aperture radar images," in *Proceedings of the International Conference on Image Processing*, 1999, pp. 857-861.

[Atte91] E. P. W. Attema, "The active microwave instrument on-board the ERS-1 satellite," *Proceedings of the IEEE*, vol. 79, pp. 791-799, June 1991.

[Bapa99] S. Bapat, "Synthesizable 200 MHZ ZBT SRAM Interface for Virtex FPGAs," Xilinx Inc., San Jose, California, Application Note XAPP136, September 1999.

[BCFH00] Digital Signal Processing Group, Rice University, "The Rice wavelet toolbox," December 2000, http://www.dsp.rice.edu/software/rwt.shtml

[Benz94] U. C. Benz, "A fuzzy block adaptive quantizer for synthetic aperture radar," in *Proceedings of the IEEE Conference on Fuzzy Systems*, 1994, pp. 1006-1011.

[BeSM95] U. Benz, K. Strodl, and A. Moreira, "A comparison of several algorithms for SAR raw data compression," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 33, pp. 1266-1276, September 1995.

[Bolle97] J. Bolle, "Coding of SAR raw data using reversible transforms," in *Proceedings of the Internation Airborne Remote Sensing Conference and Exibition*, 1997, pp. 122-129.

[Boll98] J. Bolle, "Coding of SAR raw data," presented at the European Conference on Synthetic Aperture Radar, Friedrichshafen, Germany, 1998.

[Brow67] W. M. Brown, "Synthetic aperture radar," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-3, pp. 217-229, March 1967.

[BrEK02] K. Brunham, A. El Boustani, and W. Kinsner, "A study of bit planes for the compression of raw synthetic aperture radar data," in *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, 2002, pp. 347-352.

[BuSi99] R. W. Buccigrossi and E. P. Simoncelli, "Image compression via joint statistical characterization in the wavelet domain," *IEEE Transactions on Image Processing*, vol. 8, pp. 1688-1701, December 1999.

[CLPV66] L. J. Cutrona, E. N. Leith, L. J. Porcello, and W. E. Vivian, "On the application of coherent optical processing techniques to synthetic-aperture radar," *Proceedings of the IEEE*, vol. 54, pp. 1026-1032, August 1966.

[Cook60] C. E. Cook, "Pulse compression - Key to more efficient radar transmission," *Proceedings of the IRE*, vol. 48, pp. 310-316, March 1960.

[CuHa62] L. J. Cutrona and G. O. Hall, "A comparison of techniques for achieving fine azimuth resolution," *IRE Transactions on Military Electronics*, vol. MIL-6, pp. 119-121, April 1962.

[CuMc91] J. C. Curlander and R. N. McDonough, *Synthetic Aperture Radar: Systems and Signal Processing*. New York, NY: Wiley, 1991.

[CVLH61] L. J. Cutrona, W. E. Vivian, E. N. Leith, and G. O. Hall, "A high-resolution radar combat-surveillance system," *IRE Transactions on Military Electronics*, vol. MIL-5, pp. 127-131, April 1961.

[Dans01] R. Dansereau, "Progressive image transmission using fractal and wavelet techniques with image complexity measures," Ph.D dissertation, University of Manitoba, Winnipeg, MB, Canada, 2001.

[Daub92] I. Daubechies, *Ten Lectures on Wavelets*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1992.

[DuCu92] M. Dutkiewicz and I. Cumming, "Methods of evaluating the effects of coding on SAR data," in *Proceedings of the NASA Space and Earth Science Data Compression Workshop*, 1992, pp. 59-72.

[DuCu94] M. Dutkiewicz and I. Cumming, "Evaluation of the effects of encoding on SAR data," *Photogrammetric Engineering and Remote Sensing*, vol. 60, pp. 895-904, July 1994.

[Dutk93] M. Dutkiewicz, "SAR pre-processing on-board ERS-1 data pre-conditioning," MacDonald Dettwiler, Richmond, BC, Canada, MDA Technical Report DC-TN-50-5137, ESA Technical Report ESA CR(P) 3792, 1993.

[EBJW82] C. Elachi, T. Bicknell, R. L. Jordan, and C. Wu, "Spaceborne syntheric-aperture imaging radars: Applications, techniques, and technology," *Proceedings of the IEEE*, vol. 70, pp. 1174-1209, October 1982.

[ElBK01] A. El Boustani, K. Brunham, and W. Kinsner, "A review of current raw SAR data compression techniques," in *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, 2001, pp. 925-930.

[ETHB02] A. El Boustani, A. Turiel, A. Huot, K. Brunham, and W. Kinsner, "Wavelet transform based compression techniques for raw SAR data," in *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, 2002, pp. 857-862.

[FiMW91] T. R. Fischer, M. W. Marcellin, and M. Wang, "Trellis-coded vector quantization," *IEEE Transactions on Information Theory*, vol. 37, pp. 1551-1566, November 1991.

[GeGr91] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Norwell, MA: Kluwer Academic Publishers, 1991.

[Good75] J. W. Goodman, "Statistical properties of laser speckle patterns," in *Laser Speckle and Related Phenomena*, Vol. 9, *Topics in Applied Physics*, J. C. Dainty, Ed. Berlin: Springer-Verlag, 1975.

[GoWo92] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. New York, NY: Addison-Wesley Publishing, 1992.

[Gray84] R. M. Gray, "Vector quantization," *IEEE Acoustics, Speech, and Signal Processing Magazing*, pp. 4-29, April 1984.

[GrNe98] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Transactions on Information Theory*, vol. 44, pp. 2325-2383, October 1998.

[Harr87] J. E. Harris, "Image quality measurements for comparison of synthetic aperture radar processors," M.S. thesis, Brigham Young University, Provo, Utah, USA, 1987.

[HOCC88] J. E. Harris, R. S. Ostler, D. M. Chabries, and R. W. Christiansen, "Quality measures for SAR images," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 1988, pp. 1064-1067.

[Hune89] B. L. Huneycutt, "Spaceborne imaging Radar-C instrument," *IEEE Transactions of Geoscience and Remote Sensing*, vol. 27, pp. 164-169, March 1989.

[Hune90] B. L. Huneycutt, "Innovative operating modes and techniques fot the spaceborne imageing Radar-C instrument," *IEEE Transactions of Geoscience and Remote Sensing*, vol. 28, pp. 603-608, July 1990.

[JoHW91] R. L. Jordan, B. L. Huneycutt, and M. Werner, "The SIR-C/X-SAR synthetic aperture radar system," *Proceedings of the IEEE*, vol. 79, pp. 827-838, June 1991.

[JVFC88] R. V. Jones, V. D. Vaughn, R. L. Frost, and D. M. Chabries, "The effect of codebook size on the vector quantization of SAR data," in *Proceedings of the IEEE National Radar Conference*, 1988, pp. 129-133.

[Kais94] G. Kaiser, *A Friendly Guide to Wavelets*. Boston, MA: Birkhäuser, 1994.

[Kova76] J J. Kovaly, *Synthetic Aperture Radar*. Dedham, MA: Artec House, 1976.

[KuDC94] G. Kuduvalli, M. Dutkiewicz, and I. Cumming, "Synthetic aperture radar sugnal data compression using block adaptive quantization," in *Proceedings of the Goddard Science Information Management and Data Compression Workshop*, 1994, pp. 43-57.

[KwJo89] R. Kwok and W. T. K. Johnson, "Block adaptive quantization of Magellan SAR data," *IEEE Transactions of Geoscience and Remote Sensing*, vol. 27, pp. 275-383, July 1989.

[Lebe95] D. Lebedeff, "Étude de la quantification vectorielle des données brutes issues d'un radar à synthèse d'ouverture," Ph.D dissertation, Université de Nice-Sophia Antipolis, Nice, France, 1995.

[LiBG80] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. COM-28, pp. 84-95, January 1980.

[LiBu77] R. G. Lipes and S. A. Butman, "Bandwidth compression of synthetic aperture radar imagery by quantization of raw radar data," *Proceedings of the SPIE*, vol. 119, pp. 107-114, 1977.

[Lloy82] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol IT-28, pp. 129-137, March 1982.

[LMBL95] D. Lebedeff, P. Mathieu, M. Barlaud, C. Lambert-Nebout, and P. Bellemain, "Adaptive vector quantization for raw SAR data," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 1995, pp. 2511-2514.

[Mall89] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet transform," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674-693, July 1989.

[MaOP02] E. Magli, G. Olmo, and B. Penna, "Wavelet-based compression of SAR raw data," in *Proceedings of the International Geoscience and Remote Sensing Symposium*, 2002, pp. 1129 -1131.

[MaOl02] E. Magli and G. Olmo, "Predictive coding of SAR phase history data," in *Proceedings of the International Geoscience and Remote Sensing Symposium*, 2002, pp. 1132 -1134.

[Max60] J. Max, "Quantizing for minimum distortion," *IRE Transactions on Information Theory*, pp. 7-12, March 1960.

[McCS98] I. H. McLeod, I. G. Cumming, and M. S. Seymour, "ENVISAT ASAR data reduction: Impact on SAR interferometry," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 36, pp. 589-602, March 1998.

[McCu95] I. H. McLeod and I. G. Cumming, "On-board encoding of the ENVISAT wave mode data," in *Proceedings of the International Geoscience and Remote Sensing Symposium*, 1995, pp. 1681-1683.

[Meye01] U. Meyer-Baese, *Digital Signal Processing With Field Programmable Gate Arrays*. Berlin, Germany: Springer-Verlag, 2001.

[MGBB94] J. M. Moureaux, P. Gauthier, M. Barlaud, and P. Bellemain, "Vector quantization of raw SAR data," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 1994, pp. 189-192.

[MMOP02] M. Misiti, Y. Misiti, G. Oppenheim, and J. Poggi, *Wavelet Toolbox User's Guide*, The Mathworks Inc., 2002.

[MoBl93] A. Moreira and F. Bläser, "Fusion of block adaptive and vector quantizer for efficient SAR data compression," in *Proceedings of the International Geoscience and Remote Sensing Symposium*, 1993, pp. 1583-1585.

[MuVi89] D. C. Munson and R. L. Visentin, "A signal processing view of strip-mapping synthetic aperture radar," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, pp. 2131-2147, December 1989.

[Nall01] Nallatech Limited, "Ballynuey 2 PCI Card User Guide," Nallatech Limited, Glasgow, United Kingdom, Document Number NT107-0045, October 2001.

[Olms93] C. Olmsted, "Alaska SAR Facility Scientific SAR User's Guide," Alaska SAR Facility, Fairbanks, Alaska, Technical Report ASF-SD-003, July 1993.

[OMHK99] J. W. Owens, M. W. Marcellin, B. R. Hunt, and M. Kleine, "Compression of synthetic aperture radar phase history data using trellis coded quantization techniques," in *Proceedings of the International Geoscience and Remote Sensing Symposium*, 1999, pp. 1080-1085.

[OMHK97] J. W. Owens, M. W. Marcellin, B. R. Hunt, and M. Kleine, "Compression of synthetic aperture radar phase history data using trellis coded quantization techniques," in *Proceedings of the Internation Conference on Image Processing*, 1997, pp. 592-595.

[Owen97] J. W. Owens, "Compression of synthetic aperture radar phase history data," Ph.D. dissertation, University of Arizona, Tucson, Arizona, USA, 1997.

[PaCl99] S. M. Parkes and H. L. Clifton, "The compression of raw SAR and SAR image data," *International Journal of Remote Sensing*, vol 20, pp. 3563-3581, December 1999.

[Parh99] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York, NY: John Wiley and Sons, 1999.

[Park99] S. M. Parkes, "Data compression and SpaceWire," in *Proceedings of the International Geoscience and Remote Sensing Symposium*, 1999, pp. 2267-2269.

[PaSc99] V. Pascazio and G. Schirinzi, "Wavelet transform coding for SAR raw data compression," in *Proceedings of the International Geoscience and Remote Sensing Symposium*, 1999, pp. 2251-2253.

[PaSc00] V. Pascazio and G. Schirinzi, "SAR phase history data compression by using wavelet packets," in *Proceedings of the International Geoscience and Remote Sensing Symposium*, 2000, pp. 2639-2641.

[PaSB99] V. Pascazio, G. Schirinzi, and I. D. Buttarello, "Low bit rate transform coding for SAR raw data compression, " in *Proceeding of the IEEE Radar Conference*, 1999, pp. 233-236.

[Pear79] W. A. Pearlman, "Polar quantization of a complex Gaussian random variable," *IEEE Transactions on Communications*, vol. COM-27, pp. 892-899, June 1979.

[PeSe79] W. A. Pearlman and G. H. Senge, "Optimal quantization of the Rayleigh probability distribution," *IEEE Transactions on Communications*, vol. COM-27, pp. 101-112, January 1979.

[PrFT92] W. H. Press, B. P. Flannery, and S. A. Teukolsky, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge, England: Cambridge University Press, 1988.

[RACC88] C. J. Read, D. V. Arnold, D. M. Chabries, and R. W. Christiansen, "A computation compression technique for SAR based vector quantization," in *Proceedings of the IEEE National Radar Conference*, 1988, pp. 123-128.

[RLLA91] R. K. Raney, A. P. Luscombe, E. J. Langham, and S. Ahmed, "Radarsat," *Proceedings of the IEEE*, vol. 79, pp. 839-849, June 1991.

[Sayo96] K. Sayood, *Introduction to Data Compression*. San Francisco, CA: Morgan Kaufmann, 1996.

[SBBE94] K. Strodl, U. Benz, F. Bläser, T. Eiring, and A. Moreira, "A comparison of several algorithms for on-board SAR raw data reduction," in *Proceedings of the International Geoscience and Remote Sensing Symposium*, 1994, pp. 2197-2199.

[ScSG01] C. H. Schaefer, M. Süß, and M Gottwald, "Compression of SAR raw data using the method of BABC," *Frequenz*, vol. 55, pp. 91-98, March 2001.

[Shan48] C. E. Shannon, "A mathematical theory of communications," *The Bell System Technical Journal*, vol. 27, pp. 379-423, 623-656, July, October 1948.

[ShRR62] C. W. Sherwin, J. P. Ruina, and R. D. Rawcliffe, "Some early developments in synthetic aperture radar systems," *IRE Transactions on Military Electronics*, vol. MIL-6, pp. 111-115, April 1962.

[Skol70] M. I. Skolnik, *Radar Handbook*. New York, NY: McGraw-Hill, 1970.

[StNg96] G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley, MA: Wellesley-Cambridge Press, 1996.

[Tomi78] K. Tomiyasu, "Tutorial review of synthetic-aperture radar with applications to imaging of the ocean surface," *Proceedings of the IEEE*, vol. 66, pp. 563-583, May 1978.

[VeHe92] M. Vetterl and C. Herley, "Wavelets and filter banks: theory and design," *IEEE Transactions on Signal Processing*, vol. 40, pp. 2207-2232, September 1992.

[Wils80] S. G. Wilson, "Magnitude/phase quantization of independent Gaussian variates," *IEEE Transactions on Communications*, vol. COM-28, pp. 1924-1929, November 1980.

APPENDIX A
CHARACTERISTICS OF THE RADARSAT-1 AND ERS-1 RADARS

Table A.1 Characterists of the Radarsat-1 [RLLA91] and ERS-1 [Atte91] radars.

	Radarsat-1 (Standard Mode)			**ERS-1** (Image Mode)
Radar Frequency (f_r)	5.3 GHz			5.3 GHz
Relative platform speed (V_s)	7402 m/s			7497 m/s
Platform Height (H)	800 km			785 km
Antenna height (A_H)	1.5 m			1m
Antenna length (A_L)	15 m			10 m
Beam incidence angle (θ_n)	20-49°			23°
Pulse bandwidth (B_R)	11.6	17.3	30 MHz	15.5 MHz
Sampling frequency (f_s)	12.9	18.5	32.3 MHz	18.96 MHz
Pulse repetition frequency (f_{prf})	1270-1390 Hz			1680 Hz
Time duration of pulse (τ_p)	42.0 μs			37.1μs
Ground swath width (W_g)	100 km			100 km
Azimuth resolution (δ_A)	25m			30 m
Ground range resolution (δ_{GPC})	28 m			30 m
A/D converted bits	4 bits			5 bits
I/Q wordsize	4 bits			5 bits
Compression technique	none			none
Downlink channel bandwidth	105 Mbits/s			105 Mbit/s

APPENDIX B
RAW SAR DATA TEST SETS

There are 5 raw SAR data test sets used in this thesis provided by the Alaska SAR facility. Details of the 5 original test sets and the 5 pre-processed test sets are shown in Table B.1 and Table B.2. Thumbnails using a 4x4 averting aperture of the processed amplitude images of the pre-processed test sets are shown in Fig. B.1 to Fig. B.5. Histograms of all test sets are shown in Fig. B.6 to Fig. B.16.

Table B.1 Details of the raw SAR data test sets.

Test set	Radar	Bits per sample	Complex samples per range line	Range lines
E1-22089R	ERS-1	32	5544	26624
E1-25224R	ERS-1	32	5616	20390
E2-5551R	ERS-2	32	5616	18779
R1-24576R	RADARSAT-1	16	6708	64498
R1-24919R	RADARSAT-1	16	6708	64904

Table B.2 Details of the pre-processed SAR data test sets.

Test set	Radar	Bits per sample	Spline offset	Quantization block size
E1-22089PS	ERS-1	256	0.08	1386
E1-25224PS	ERS-1	256	0.08	702
E2-5551PS	ERS-2	256	0.08	702
R1-24576PS	RADARSAT-1	256	0.3	3354
R1-24919PS	RADARSAT-1	256	0.3	3354

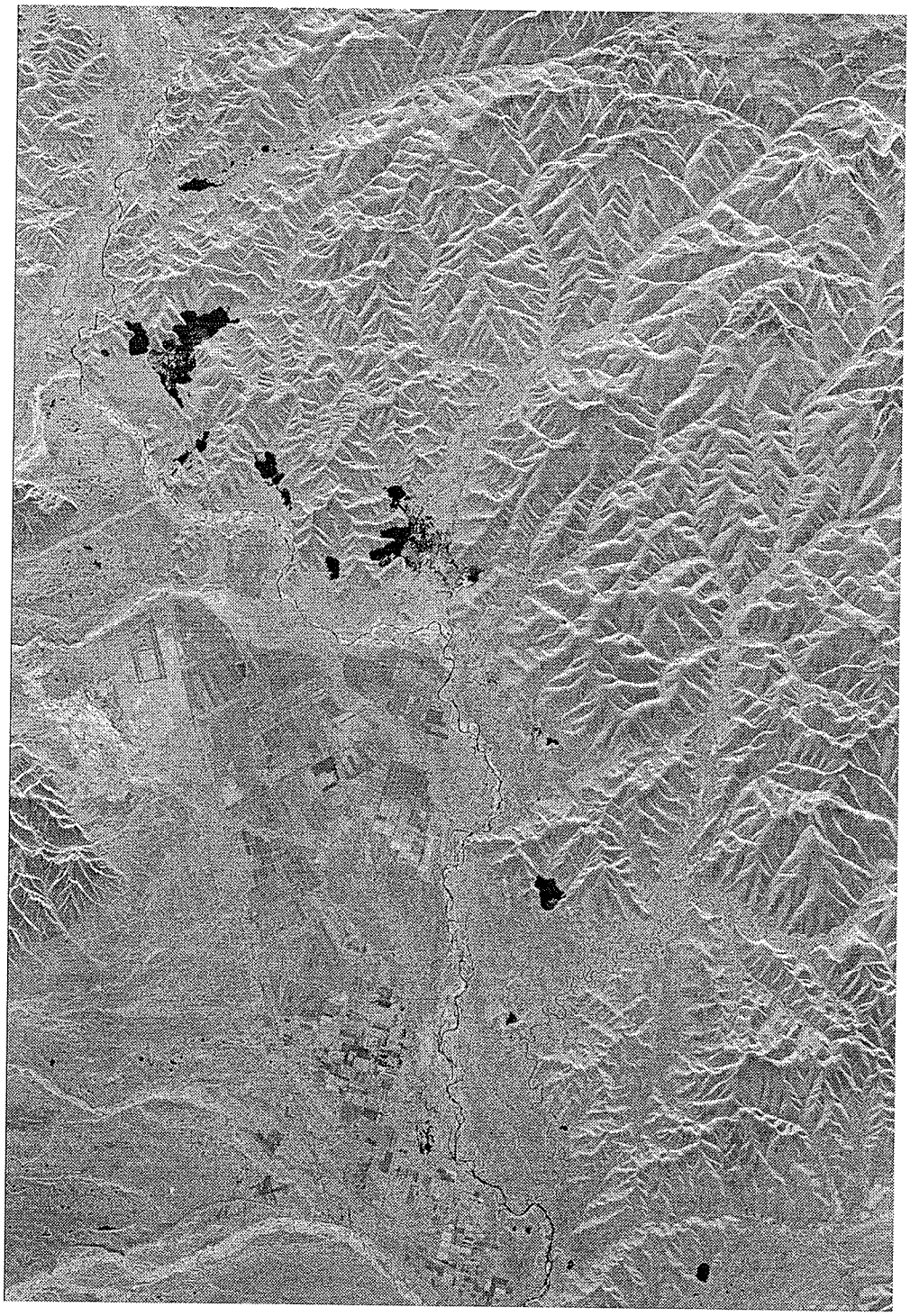


Fig. B.1. Processed amplitude image of E1-22089PS data set.

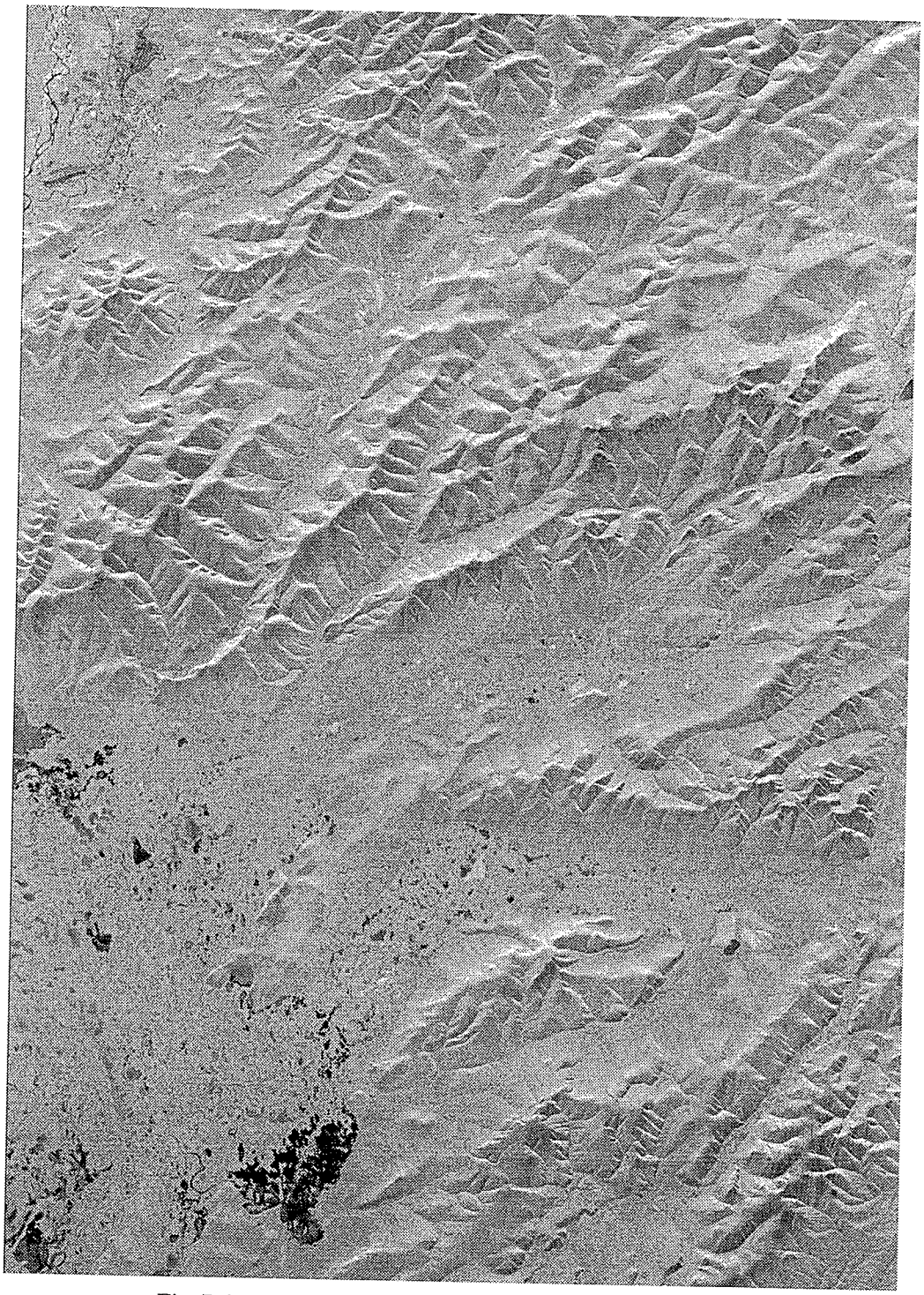


Fig. B.2. Processed amplitude image of E1-25224PS data set.

Fig. B.3. Processed amplitude image of E2-5551PS data set.

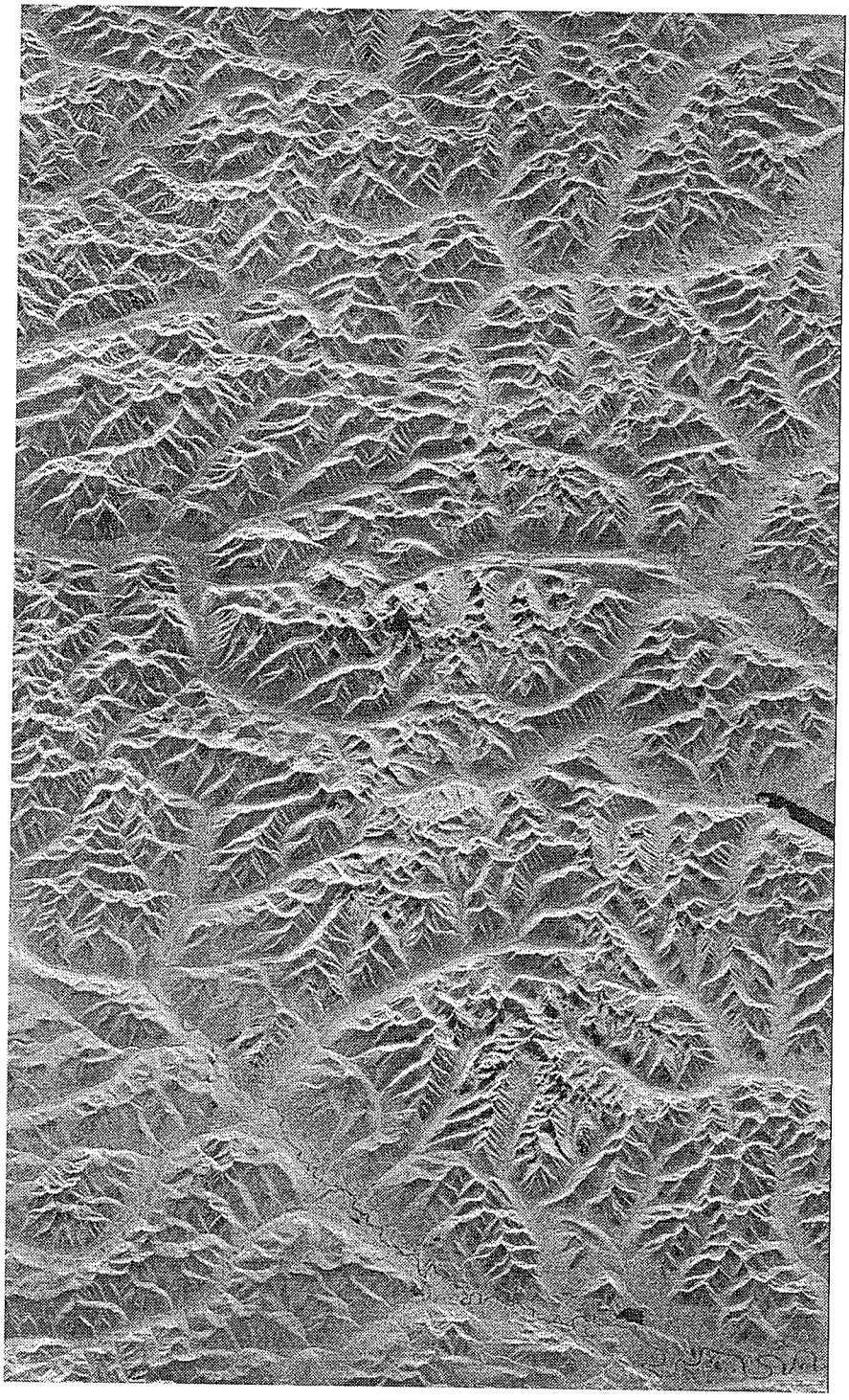


Fig. B.4. Processed amplitude image of R1-24576PS data set.

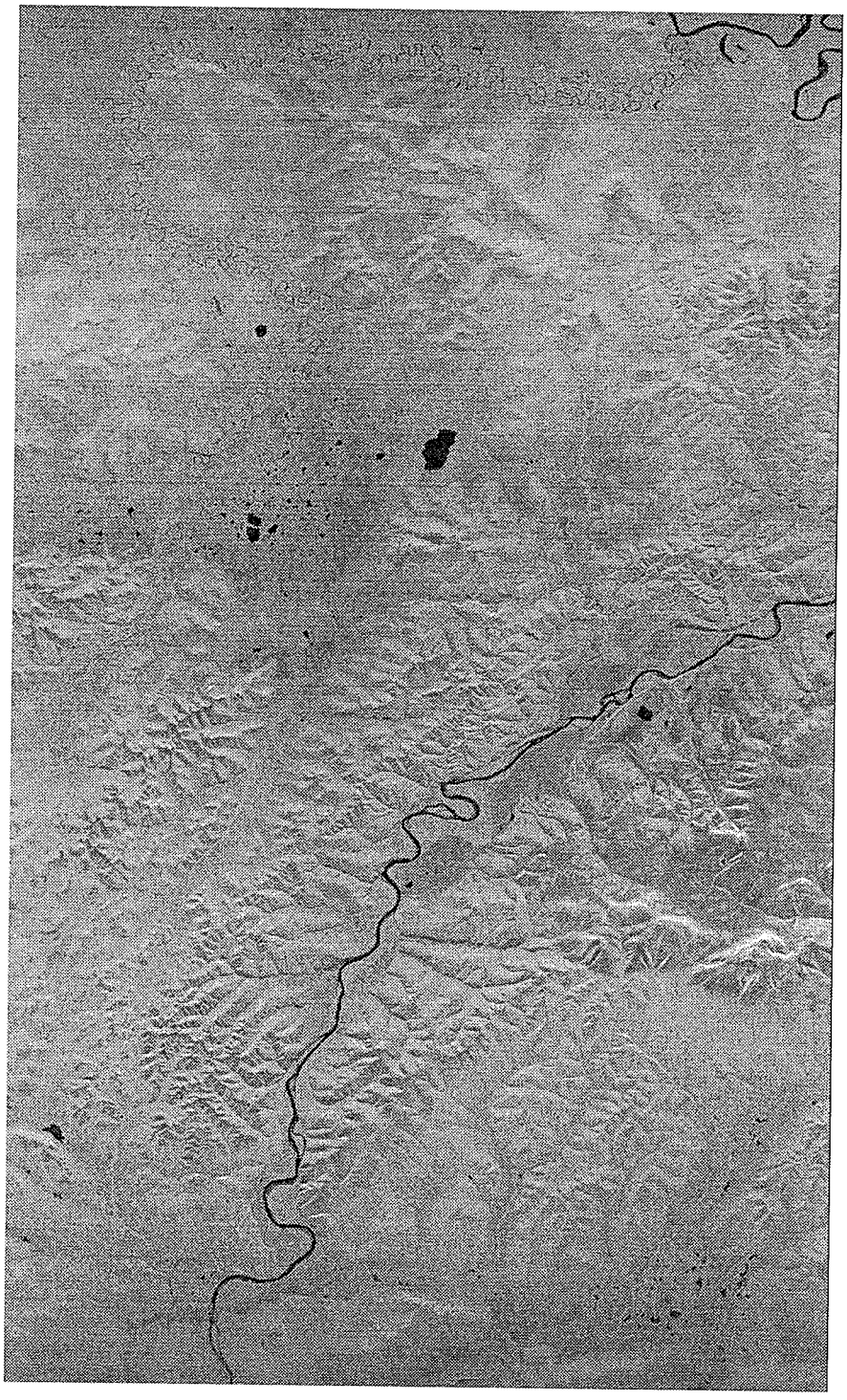


Fig. B.5. Processed amplitude image of R1-24919PS data set.

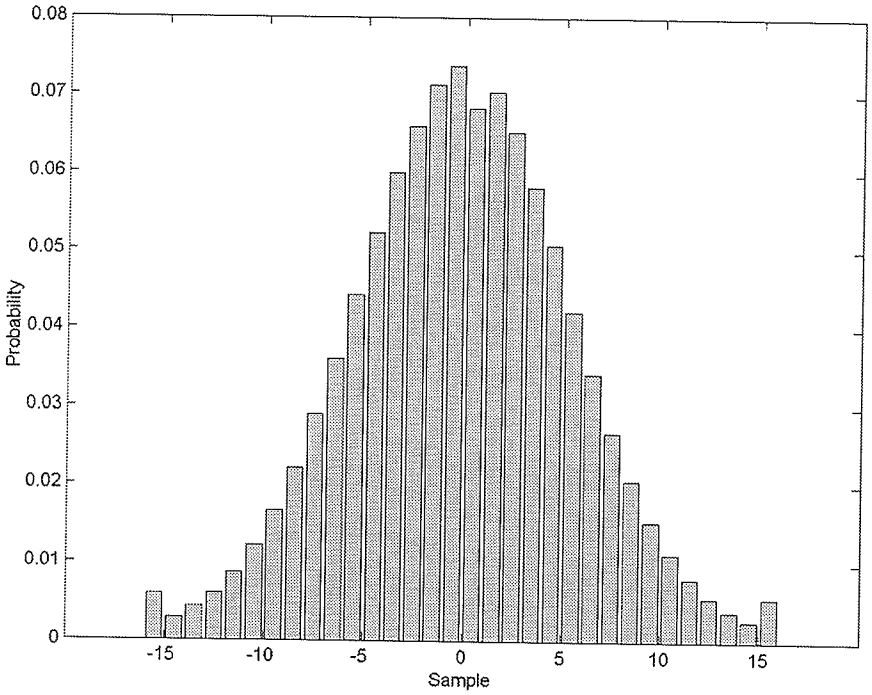


Fig. B.6. Histogram of E1-22089R dataset.

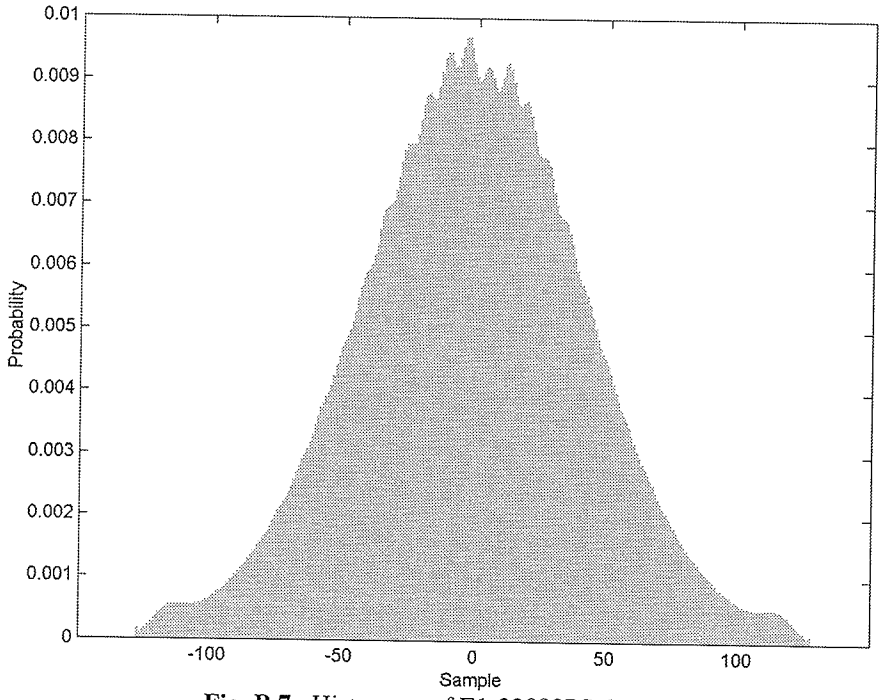


Fig. B.7. Histogram of E1-22089PS dataset.

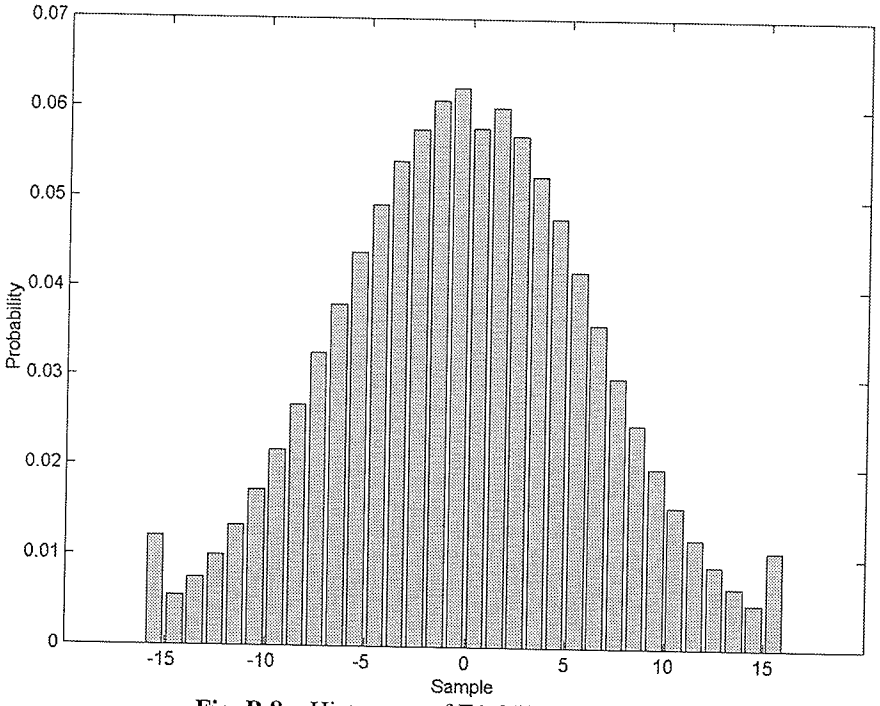


Fig. B.8. .Histogram of E1-25224R dataset.

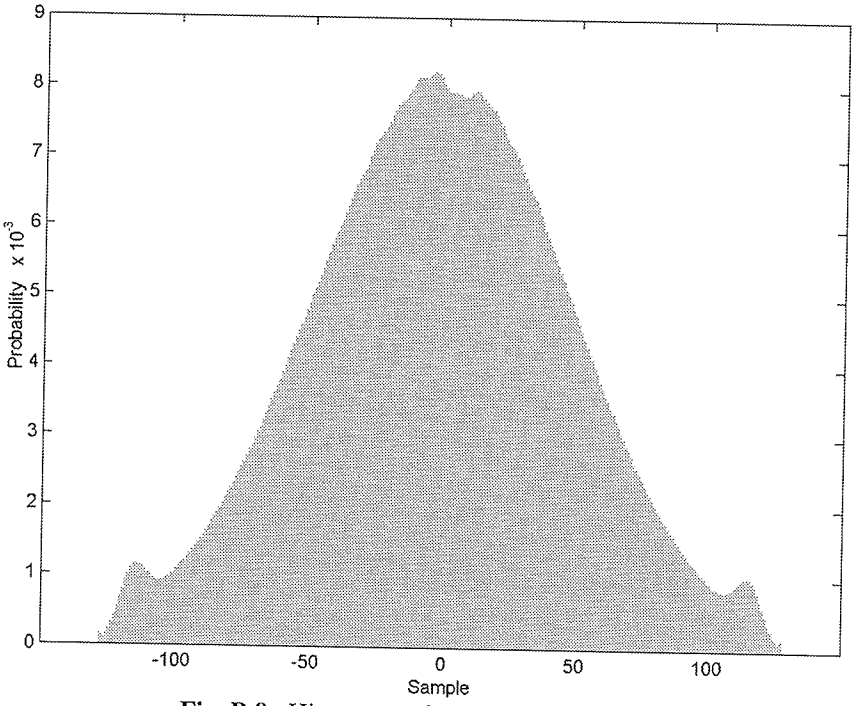


Fig. B.9. Histogram of E1-25224PS dataset.

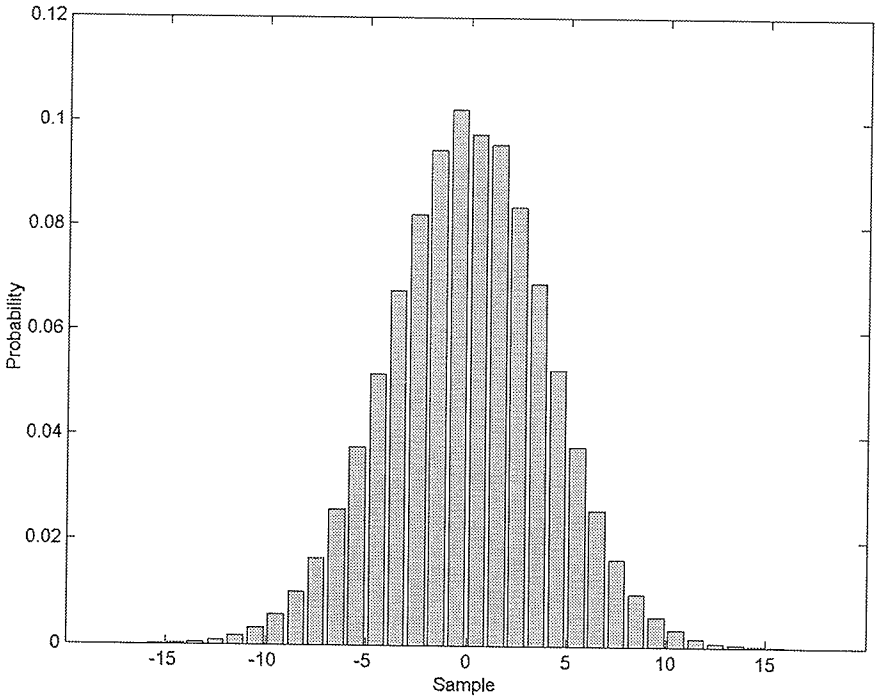


Fig. B.10. Histogram of E2-5551R dataset.

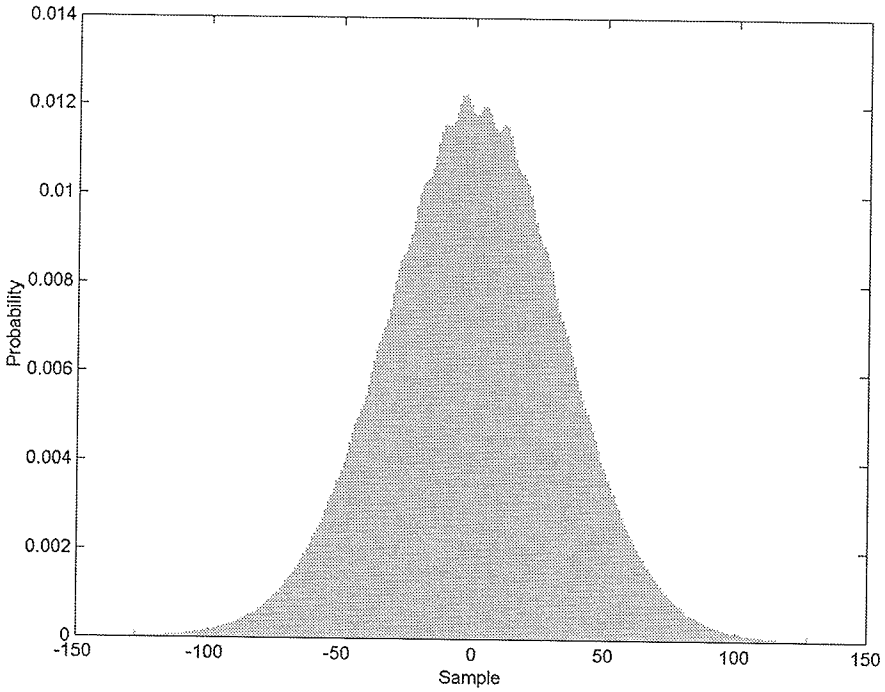


Fig. B.11. Histogram of E2-5551PS dataset.

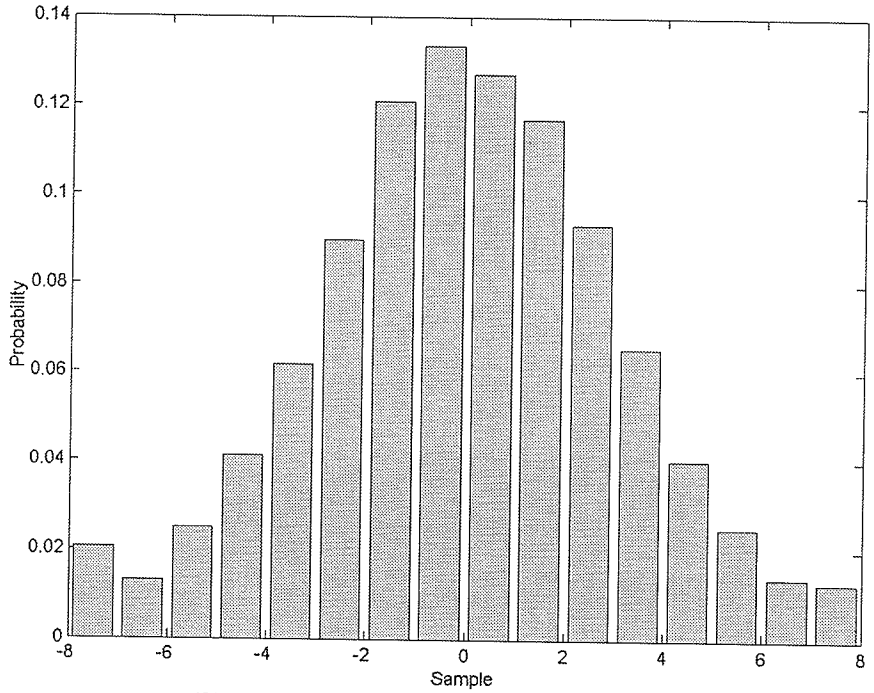


Fig. B.12. Histogram of R1-24576R dataset.

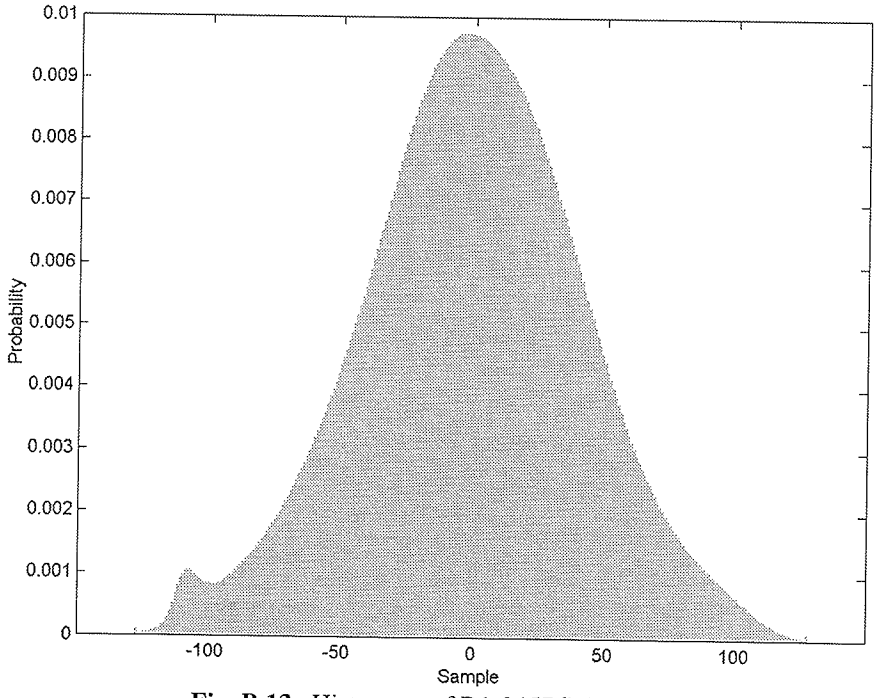


Fig. B.13. Histogram of R1-24576PS dataset.

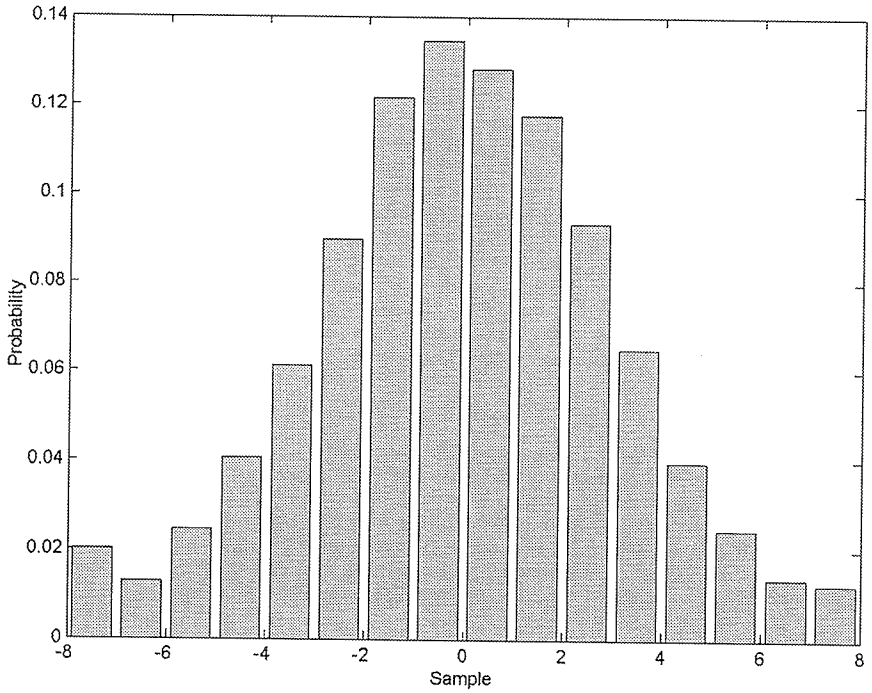


Fig. B.14. Histogram of R1-24919R dataset.

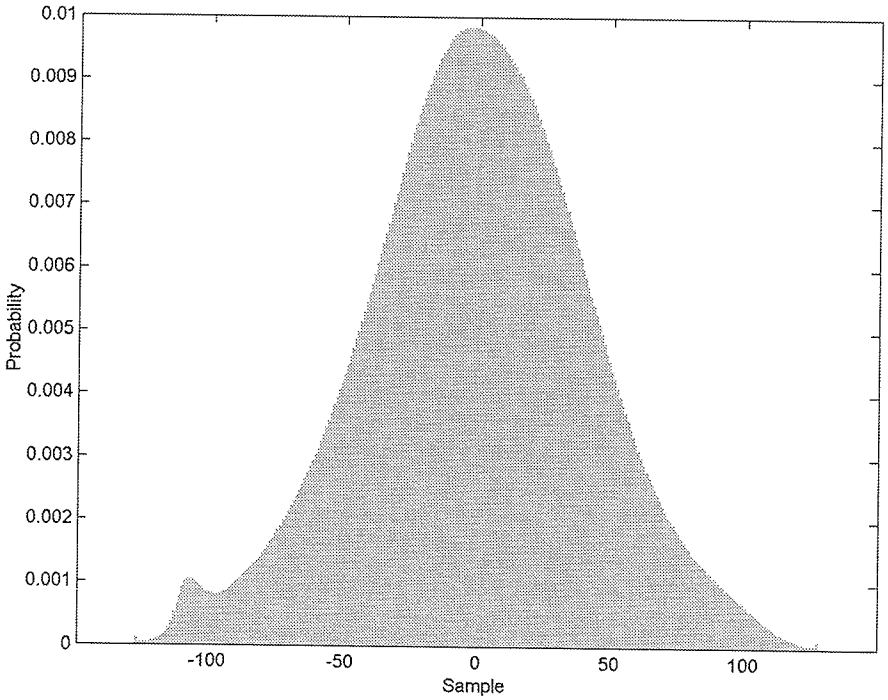


Fig. B.15. Histogram of R1-24919PS dataset.

Table B.3 Statistics of the original and cubic spline pre-processed data from the E1-22089R test set.

Statistic	Original	Pre-processed
Entropy	4.52	7.47
Redundancy	0.48	0.53
Standard deviation	16.37	133.68
σ factor		8.17
Kurtosis	1.44	1.39
Skewness	1.16	1.14
Mean	0.001	0.005

Table B.4 Statistics of the original and cubic spline pre-processed data from the E1-25224R test set.

Statistic	Original	Pre-processed
Entropy	4.69	7.62
Redundancy	0.31	0.38
Standard deviation	16.61	135.15
σ factor		8.14
Kurtosis	1.55	1.48
Skewness	1.20	1.17
Mean	0.001	0.01

Table B.5 Statistics of the original and cubic spline pre-processed data from the E2-5551R test set.

Statistic	Original	Pre-processed
Entropy	4.04	7.11
Redundancy	0.96	0.89
Standard deviation	16.01	132.04
σ factor		8.25
Kurtosis	1.24	1.25
Skewness	1.09	1.09
Mean	0.001	0.01

Table B.6 Statistics of the original and cubic spline pre-processed data from the R1-24576R test set.

Statistic	Original	Pre-processed
Entropy	3.64	7.43
Redundancy	0.36	0.57
Standard deviation	8.06	133.84
σ factor		16.60
Kurtosis	1.53	1.37
Skewness	1.19	1.13
Mean	0.00	0.00

Table B.7 Statistics of the original and cubic spline pre-processed data from the R1-24919R test set.

Statistic	Original	Pre-processed
Entropy	3.63	7.42
Redundancy	0.37	0.58
Standard deviation	8.06	133.77
σ factor		16.60
Kurtosis	1.52	1.36
Skewness	1.19	1.13
Mean	0.00	0.00

APPENDIX C
MATLAB CODE

C.1 Pseduosimulated Processor (resampleCCSDspline.m)

```
%resampleCCSDspline Resamples the dynamic range quantization on
%SAR computer capable signal data.
%[CCSDout] = RESAMPLECCSDspline(filename,filestruct,
%origbits,numbits,offset,debug) creates a file called
%FILENAME.RESAMPSPLINE.CCSD containing alternating I/Q samples where
%data resampled to NUMBITS from ORIGBITS using cubic spline
%interpolation and an offset of OFFSET.
%
%   FILESTRUCT is a structure of
%   [FILEHEADER RECHEADER SAMPLES_PER_REC NUMREC].
%   The inputfile FILNAME will be read using a file header of FILEHEADER
%bytes, a and a record length of RECHEADER + 2*SAMPLES_PER_REC.
%
%
%   Enter a value of 1 for DEBUG if you wish to enter debug mode. This
%will enable data [CCSDout] output.
%
%

function [CCSDout, dataout] =
resampleCCSDspline(filename,filestruct,origbits,numbits,offset,debug)


fileheader = filestruct(1);
recheader = filestruct(2);
samples_per_rec = filestruct(3);
numrec = filestruct(4);

debugmode = 0;
if nargin == 6
    if debug == 1
        debugmode = 1;
        CCSDout = zeros(2*samples_per_rec*numrec,1);
        dataout = zeros(2*samples_per_rec*numrec,1);
    end
end
if ~ ((numbits > 0) & (numbits <= 8))
    ['NUMBITS must be between 1 and 8 bits']
    return;
end

if ((numbits-origbits-1) < 0)
    ['NUMBITS must be greater then ORIGBITS']
```

```
    return;
end

CCSD_filename = [filename,'.CCSD'];
CCSD_file = fopen(CCSD_filename,'r');

if (CCSD_file == -1)                                    % Is file ASF
CCSD File?
    CCSD_filename = [filename,'.D'];
    CCSD_file = fopen(CCSD_filename,'r');
    if (CCSD_file == -1)
        CCSD_filename = [filename,'.raw'];
        CCSD_file = fopen(CCSD_filename,'r');
        if (CCSD_file == -1)
            ['Could not open CCSD file']
            return
        else
        resamp_filename = [filename,'.RESAMPSPLINE.raw'];
        copyfile([filename,'.in'],[filename,'.RESAMPSPLINE.in']);
        copyfile([filename,'.fmt'],[filename,'.RESAMPSPLINE.fmt']);
        end
    else
        resamp_filename = [filename,'.RESAMPSPLINE.D'];
        copyfile([filename,'.L'],[filename,'.RESAMPSPLINE.L']);
    end
else
    resamp_filename = [filename,'.RESAMPSPLINE.CCSD'];
end


fseek(CCSD_file,0,'bof');

div_file = fopen(resamp_filename,'w');

headerdata = fread(CCSD_file,fileheader,'uchar');
fwrite(div_file,headerdata,'uchar');

parts = linspace(0,1,2^numbits+1);

for i = 1:numrec
     if (mod(i,100) == 0)
        ['Processing Line ',num2str(i),' of ',num2str(numrec)]
    end

    headerdata = fread(CCSD_file,recheader,'uchar');
    fwrite(div_file,headerdata,'uchar');

    recdata = fread(CCSD_file,2*samples_per_rec,'uchar');
    outdata = recdata;

    recdata = recdata-(2^origbits-1)/2;
```

```
    x = 1:1:2*samples_per_rec;
    newdata = spline(x,recdata,x+offset);

    newdata = newdata-min(newdata);
    newdata = newdata./max(newdata);


    recdata = quantiz(newdata,parts(2:length(parts)-1));

    if debugmode == 1
CCSDout(((i-1)*2*samples_per_rec+1):(i*2*samples_per_rec)) = recdata;
dataout(((i-1)*2*samples_per_rec+1):(i*2*samples_per_rec)) = outdata;
    end


    fwrite(div_file,recdata,'uchar');
end

fclose(CCSD_file);
fclose(div_file);
```

C.2 MRA Wavelet Analysis (MRA_analysis.m)

```
%MRA_analysis performs a 1D MRA transform.
%[a1,d1] = MRA_anlysis(a0,wavelet) performs the multiresolution
analysis %transform on A0 usign the wavelet WAVELET. The function
returns the %approximation A1 and details D1, at the next resolution.
%

function [a1,d1] = MRA_anlysis(a0,wavelet)

size_a0 = length(a0);
if (mod(size_a0,2) ~= 0)
    ['a0 must have dimensions of a power of 2']
    return;
end

[h,g] = wavelet_filters(wavelet,'analysis');
if (length(h) == 0)
    ['Wavelet ',wavelet,' unknown']
end

a1 = wave_conv(a0,h,'analysis');
d1 = wave_conv(a0,g,'analysis');
```

C.3 MRA Wavelet Synthesis (MRA_synthesis.m)

```
%MRA_synthesis performs a 1D MRA reconstruction.
%[A0] = MRA_synthesis(a1,d1,wavelet) performs the multiresolution
%analysis reconstruction on A1 and D1 usign the wavelet WAVELET. The
%function returns the approximation A0, at the next resolution.
%

function [a0] = MRA_synthesis(a1,d1,wavelet)


size_a1 = size(a1);
size_d1 = size(d1);

if (size_a1 ~= size_d1)
    ['a1 and d1 must have the same dimensions']
    return;
end

[h,g] = wavelet_filters(wavelet,'synthesis');
if (length(h) == 0)
    ['Wavelet ',wavelet,' unknown']
end


a0 = wave_conv(a1,h,'synthesis');
d0 = wave_conv(d1,g,'synthesis');

a0 = a0+d0;
```

C.4 Wavelet Convolution (wave_conv.m)

```
%wave_conv performs a 1D convolution using a specified wavelet
filter.
%wave_conv(sample,filter,type) convolves SAMPLE with FILTER and
returns %the result RES. The RES returned will always start with
indicies 0 and %end at an index (sample_len). The value of TYPE
specifies with synthesis %or analysis which will determine if h0 is on
the left or right of %filter. This value of TYPE also specifies
whether RES should be %upsampled by 2 or downsampled by 2.

function [res] = wave_conv(sample, filter,type)

filter_len = length(filter);
sample_size = size(sample);

if (strcmpi(type,'analysis') == 1) %wavelet decomposition (h0 on
right)
    res = w_convr(sample,filter);

    if (mod(filter_len,2) == 0)
        res = res(filter_len:2:length(res));
    else
        res = res(filter_len-1:2:length(res));
    end

elseif (strcmpi(type,'synthesis') == 1) % (h0 on left side of filter)
    sample_size = size(sample);
    if (sample_size(1) >= sample_size(2))
        uped_sample = zeros(2*sample_size(1),1);
    else
        uped_sample = zeros(1,2*sample_size(2));
    end

    uped_sample(1:2:2*length(sample)) = sample;

    res = w_convr(uped_sample,filter);
    res = res(1:2*length(sample));


else
    error(['Error in type ',type,' unknown']);
end
```

C.5 Convolution (w_convr.m)

```
%w_convr performs a 1D convolution
%w_convr(sample,filtere) convolves SAMPLE with FILTER and returns the
%result RES.

function [res] = w_conv(sample, filter)

sample_len = length(sample);
filter_len = length(filter);

sample_size = size(sample);
filter_size = size(filter);

if (sample_size(1) > sample_size(2))
    res = zeros(sample_len+filter_len-1,1);
    series_head = sample_len-1:-1:sample_len-(filter_len-1);
    series_head = mod(series_head,sample_len)+1;
    series_head = series_head(filter_len-1:-1:1);
    series_tail = sample_len:sample_len+(filter_len-1)-1;
    series_tail = mod(series_tail,sample_len)+1;
sample = [sample(series_head); sample; sample(series_tail)];

    if (filter_size(2) > filter_size(1))
        filter = filter';
    end
else
    res = zeros(1,sample_len+filter_len-1);
series_head = sample_len-1:-1:sample_len-(filter_len-1);
    series_head = mod(series_head,sample_len)+1;
    series_head = series_head(filter_len-1:-1:1);
    series_tail = sample_len:sample_len+(filter_len-1)-1;
    series_tail = mod(series_tail,sample_len)+1;

    sample = [sample(series_head) sample sample(series_tail)];

    if (filter_size(1) > filter_size(2))
        filter = filter';
    end
end
res_len = sample_len+filter_len-1;

for i = 1:res_len
    res(i) = sum(sample(i:i+filter_len-1).*filter(filter_len:-1:1));
end
```

C.6 Wavelet Filters (wavelet_filtes.m)

```
%wavelet_filters returns the appropriate wavelet filters.
%[h,g] = wavelet_filters(wavelet) returns the high pass wavelet
filter G %and the low pass filter H for the wavelet WAVELET. The
decomposition or
%reconstruction filters can be chosen by setting TYPE to:
%       'analysys'      : decomposition filter
%       'synthesis'     : reconstruction filter

function [h,g] = wavelet_filters(wavelet,type);

wavelet = deblank(wavelet);
[ld, hd, lr, hr] = wfilters(wavelet);

%=========================================================================
===%=                                        haar Wavelet
%=========================================================================
===if (strcmpi(wavelet,'haar') == 1)
    if (strcmpi(type,'synthesis') == 1)
        h = zeros(length(ld),1);
        h(1) = 0.70710678118654757;
        h(2) = 0.70710678118654757;

        g = zeros(length(hd),1);
        g(1) = 0.70710678118654757;
        g(2) = -0.70710678118654757;

    elseif (strcmpi(type,'analysis') == 1)
        h = zeros(length(ld),1);
        h(1) = 0.70710678118654757;
        h(2) = 0.70710678118654757;


        g = zeros(length(hd),1);
        g(1) = -0.70710678118654757;
        g(2) = 0.70710678118654757;
    end
%=========================================================================
===%=                                        db2 Wavelet
%=========================================================================
===elseif (strcmpi(wavelet,'db2') == 1)
    if (strcmpi(type,'synthesis') == 1)
        h = zeros(length(ld),1);
        h(1) = 0.48296291314469025;
        h(2) = 0.83651630373746899;
        h(3) = 0.22414386804185735;
        h(4) = -0.12940952255092145;

        g = zeros(length(hd),1);
```

```
        g(1) = -0.12940952255092145;
        g(2) = -0.22414386804185735;
        g(3) = 0.83651630373746899;
        g(4) = -0.48296291314469025;

    elseif (strcmpi(type,'analysis') == 1)
        h = zeros(length(ld),1);
        h(1) = -0.12940952255092145;
        h(2) = 0.22414386804185735;
        h(3) = 0.83651630373746899;
        h(4) = 0.48296291314469025;


        g = zeros(length(hd),1);
        g(1) = -0.48296291314469025;
        g(2) = 0.83651630373746899;
        g(3) = -0.22414386804185735;
        g(4) = -0.12940952255092145;
    end
```

Repeated for all wavlets

```
else
    error(['Error! Wavelet ',wavelet,' not found'])
end
```

C.7 Error Due To Coefficient Quantization (MRA_error.m)

```
%MRA_error calculates the error after a MRA decomposition to a
%specific level (j) and reconstruction by quantizing the
%coefficients.
%
%[a0_r] = MRA_error(a0,level, numbits, wavelet) reconstructs A0 from
%level LEVEL using MRA and the wavelet WAVELET using the coefficients
%quantized using NUMBITS. a0_r is the reconstruction.
%
%   If the level is specified as -1, the maximum level is descended.

function [a0_r] = MRA_error(a0,level, numbits, wavelet)

length_a0 = length(a0);

if ((level > floor(log2(length_a0))) | (level < 0))
    level = floor(log2(length_a0));
end

[S] = MRA_decomp(a0,level,wavelet);


mean_vec = zeros(level+1,1);
var_vec = zeros(level+1,1);

S_len = length(S);

%% Phase 1: Calculation of Mean and standard deviation of each
coefficient block.

pos = 0;
for i = 1:level
    Aj_len = length(S)/2^i;
    mean_vec(i) = mean(S(pos+1:pos+Aj_len));
    var_vec(i) = std(S(pos+1:pos+Aj_len));
    pos = pos+Aj_len;
end

mean_vec(level+1) = mean(S(pos+1:pos+Aj_len));
var_vec(level+1) = std(S(pos+1:pos+Aj_len));

%% Phase 2: Quantization of coefficients

Sr = zeros(size(S));

[Partition Codebook] = MaxLloyd(2^numbits);
Partition = Partition(2:2^numbits)';
Codebook = Codebook';
```

```
pos = 0;
for i = 1:level
    Aj_len = length(S)/2^i;
    if (var_vec(i) ~= 0)
        [comp_index comp_decomp_data] = quantiz(S(pos+1:pos+Aj_len)-
mean_vec(i),Partition.*var_vec(i),Codebook.*var_vec(i));
        Sr(pos+1:pos+Aj_len) = comp_decomp_data + mean_vec(i);
    else
        Sr(pos+1:pos+Aj_len) = S(pos+1:pos+Aj_len);
    end
    pos = pos+Aj_len;
end

if (var_vec(level+1) ~= 0)
    [comp_index comp_decomp_data] = quantiz(S(pos+1:pos+Aj_len)-
mean_vec(level+1),Partition.*var_vec(level+1),Codebook.*var_vec(level
+1));
    Sr(pos+1:pos+Aj_len) = comp_decomp_data + mean_vec(level+1);
else
    Sr(pos+1:pos+Aj_len) = mean_vec(level+1);
end

a0_r = MRA_recon(Sr,level,wavelet);
```

C.8 Block Adaptive Quantization (baqCCSD.m)

```
%baqCCSD Performs a block adaptive quantization on 8-bit SAR computer
%capable signal data.
%[CCSDout] = BAQCCSD(FILENAME,FILESTRUCT,ABLOCK,RBLOCK,NUMBITS,DEBUG)
%   creates a file called FILENAME.BAQ.CCSD containing alternating
%   I/Q samples where data is compressed to NUMBITS (then
    decompressed) using BAQ.
%
%    FILESTRUCT is a structure of
%           [FILEHEADER RECHEADER SAMPLES_PER_REC NUMREC].
%   The inputfile FILNAME will be read using a file header of
%   FILEHEADER bytes, a and a record
%   length of RECHEADER + 2*SAMPLES_PER_REC.
%
%   The BAQ blocksize is ABLOCK samples in the Azimuth direction and
%   RBLOCK samples in the range direction.
%
%   Enter a value of 1 for DEBUG if you wish to enter debug mode. This
%   will enable data [CCSDout] output.
%
%

function
[CCSDout] = baqCCSD(filename,filestruct,ablock,rblock,numbits,debug)

t1 = cputime;

fileheader = filestruct(1);
recheader = filestruct(2);
samples_per_rec = filestruct(3);
numrec = filestruct(4);

debugmode = 0;
if nargin == 6
    if debug == 1
        debugmode = 1;
        CCSDout = zeros(2*samples_per_rec*numrec,1);
    end
end
if ~ ((numbits > 0) & (numbits <= 8))
    ['NUMBITS must be between 1 and 8 bits']
    return;
end
if (mod(samples_per_rec*2,rblock) ~= 0)
    ['Blocksize must be multiple of dataset']
    return;
end
```

```
CCSD_filename = [filename,'.CCSD'];
CCSD_file = fopen(CCSD_filename,'r');
if (CCSD_file == -1)                                    % Is file ASF
CCSD File?
    CCSD_filename = [filename,'.D'];
    CCSD_file = fopen(CCSD_filename,'r');
    if (CCSD_file == -1)
        CCSD_filename = [filename,'.raw'];
        CCSD_file = fopen(CCSD_filename,'r');
        if (CCSD_file == -1)
            ['Could not open CCSD file']
            return
        else
        baq_filename = [filename,'.BAQ_',num2str(numbits),'.raw'];
copyfile([filename,'.in'],[filename,'.BAQ_',num2str(numbits),'.in']);
copyfile([filename,'.fmt'],[filename,'.BAQ_',num2str(numbits),'.fmt']
);
        end
    else
        baq_filename = [filename,'.BAQ_',num2str(numbits),'.D'];

copyfile([filename,'.L'],[filename,'.BAQ_',num2str(numbits),'.L']);
    end
else
    baq_filename = [filename,'.BAQ_',num2str(numbits),'.CCSD'];
end

baq_file = fopen(baq_filename,'w');

%Write File Header
headerdata = fread(CCSD_file,fileheader,'uchar');
fwrite(baq_file,headerdata,'uchar');


% PASS 1 : Calculate Mean Matrix
['PASS 1 : Calculate Mean Matrix']

mean_mat = zeros(ceil(numrec/ablock),2*samples_per_rec/rblock);
block_mat = ones(ceil(numrec/ablock),2*samples_per_rec/
rblock).*(rblock*ablock);
block_mat(ceil(numrec/ablock),:) =  rblock.*(numrec-
ablock.*floor(numrec/ablock));
```

```
% PASS 2 : Calculate Standard Deviation Matrix
['PASS 2 : Calculate Standard Deviation Matrix']
SD_mat = zeros(ceil(numrec/ablock),2*samples_per_rec/rblock);
fseek(CCSD_file,fileheader,'bof');

for i = 1:numrec
    fseek(CCSD_file,recheader,'cof');
    for j = 1:samples_per_rec*2/rblock
        recdata = fread(CCSD_file,rblock,'uchar') - 127.5;
        recdata = recdata - mean_mat(floor((i-1)/ablock)+1,j);
        SD_mat(floor((i-1)/ablock)+1,j) = sum([SD_mat(floor((i-1)/
ablock)+1,j); recdata.^2]);
    end
end

SD_mat = sqrt(SD_mat./(block_mat-1));


% PASS 3 : Perform Compression/Decompression
['PASS 3 : Perform Compression/Decompression']

[Partition reconstruction_base] = MaxLloyd(2^numbits);
Partition = Partition(2:2^numbits)';
reconstruction_base = reconstruction_base';


fseek(CCSD_file,fileheader,'bof');
for i = 1:numrec
    headerdata = fread(CCSD_file,recheader,'uchar');
    fwrite(baq_file,headerdata,'uchar');
    for j = 1:samples_per_rec*2/rblock
        recdata = fread(CCSD_file,rblock,'uchar') - 127.5;
       [comp_index comp_decomp_data] =
       quantiz(recdata,Partition.*SD_mat(floor((i-1)/
ablock)+1,j),reconstruction_base.*SD_mat(floor((i-1)/ablock)+1,j));
        fwrite(baq_file,floor(comp_decomp_data+127.5),'uchar');

       if debugmode == 1
            CCSDout((((i-1)*samples_per_rec*2)+(j-1)*rblock+1):(((i-
1)*2*samples_per_rec)+j*rblock)) = floor(comp_decomp_data'+127.5);
        end
    end
end



fclose(CCSD_file);
fclose(baq_file);

t2 = cputime;
```

C.9 1-D Wavelet BAQ (wavebaqCCSD.m)

```
%waveletbaqCCSD Performs a wavelet domain block adaptive quantization
%on 8-bit SAR computer capable signal data.
%   [CCSDout] =
%   WAVELETBAQCCSD(FILENAME,FILESTRUCT,RBLOCK,NUMBITS,
%   WAVELET,NUMLEVEL,DEBUG)
%   creates a file called FILENAME.WAVBAQ.CCSD containing alternating
%   I/Q samples where data is compressed to NUMBITS (then
%   decompressed) using 1-D Wavelet BAQ.
%
%   FILESTRUCT is a structure of
%          [FILEHEADER RECHEADER SAMPLES_PER_REC NUMREC].
%   The inputfile FILNAME will be read using a file header of
%   FILEHEADER bytes, a and a record
%   length of RECHEADER + 2*SAMPLES_PER_REC.
%
%   The BAQ blocksize is RBLOCK samples in the range direction.
%
%   Enter a value of 1 for DEBUG if you wish to enter debug mode. This
%   will enable data [CCSDout] output.
%
%

function [CCSDout, Rawout] =
       waveletbaqCCSD(filename,filestruct,rblock,numbits,
       wavelet,numlevel,debug)

t1 = cputime;
fileheader = filestruct(1);
recheader = filestruct(2);
samples_per_rec = filestruct(3);
numrec = filestruct(4);

debugmode = 0;
if nargin == 7
    if debug == 1
        debugmode = 1;
        CCSDout = zeros(2*samples_per_rec*numrec,1);
        Rawout = zeros(2*samples_per_rec*numrec,1);
    end
end
if ~ ((numbits > 0) & (numbits <= 8))
    ['NUMBITS must be between 1 and 8 bits']
    return;
end
if (mod(samples_per_rec*2,rblock) ~= 0)
    ['Blocksize must be multiple of dataset']
    return;
end
```

```
CCSD_filename = [filename,'.CCSD'];
CCSD_file = fopen(CCSD_filename,'r');
if (CCSD_file == -1)                                    % Is file ASF
CCSD File?
    CCSD_filename = [filename,'.D'];
    CCSD_file = fopen(CCSD_filename,'r');
    if (CCSD_file == -1)
        CCSD_filename = [filename,'.raw'];
        CCSD_file = fopen(CCSD_filename,'r');
        if (CCSD_file == -1)
            ['Could not open CCSD file']
            return
        else
       baq_filename = [filename,'.WAVEBAQ_',num2str(numbits),'.raw'];
copyfile([filename,'.in'],[filename,'.WAVEBAQ_',num2str(numbits),'.in
']);

copyfile([filename,'.fmt'],[filename,'.WAVEBAQ_',num2str(numbits),'.f
mt']);
        end
    else
        baq_filename = [filename,'.WAVEBAQ_',num2str(numbits),'.D'];

copyfile([filename,'.L'],[filename,'.WAVEBAQ_',num2str(numbits),'.L']
);
    end
else
    baq_filename = [filename,'.WAVEBAQ_',num2str(numbits),'.CCSD'];
end

baq_file = fopen(baq_filename,'w');


%Write File Header
headerdata = fread(CCSD_file,fileheader,'uchar');
fwrite(baq_file,headerdata,'uchar');

for i = 1:numrec
    headerdata = fread(CCSD_file,recheader,'uchar');
    fwrite(baq_file,headerdata,'uchar');
    for j = 1:samples_per_rec*2/rblock
        recdata = fread(CCSD_file,rblock,'uchar') - 127.5;

        a0_r = MRA_error(recdata,numlevel,numbits,wavelet);
        fwrite(baq_file,floor(a0_r+127.5),'uchar');
```

```
if debugmode == 1
            CCSDout((((i-1)*samples_per_rec*2)+(j-1)*rblock+1):(((i-
1)*2*samples_per_rec)+j*rblock)) = floor(a0_r'+127.5);
            Rawout((((i-1)*samples_per_rec*2)+(j-1)*rblock+1):(((i-
1)*2*samples_per_rec)+j*rblock)) = recdata+127.5;
        end
    end
end



fclose(CCSD_file);
fclose(baq_file);

t2 = cputime;
```

APPENDIX D
VHDL CODE

D.1 Block Adaptive Quantization Core (baq_core.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity BAQ_Core is
    Generic (
       BlockSize        : integer := 702;
       AccumulatorWidth : integer := 17;
       DataWidth        : integer := 8;
       MagStatDef  : integer := 0;
       DefaultSigma     : integer := 45
    );
    Port (
       DataIn : in std_logic_vector(DataWidth-1 downto 0);
       DataEmpty : in std_logic;

       DataOut : out std_logic_vector(1 downto 0);
       DataReady : out std_logic;
       StallPipe : in std_logic;
       MagStat : out std_logic_vector(AccumulatorWidth-1 downto 0);

       LUT_AddrBus : out std_logic_vector(16 downto 0);
       LUT_DataBus : in std_logic_vector(31 downto 0);

       CLK : in std_logic;
       Reset : in std_logic
    );
end BAQ_Core;

architecture Behavioral of BAQ_Core is

signal Threshold : unsigned(DataWidth-1 downto 0);

signal Pipe3Enable, Pipe4Enable, Pipe5Enable : std_logic;
signal Pipe3Data, Pipe4Data, Pipe5Data : signed(DataWidth-1 downto 0);

signal DataLoad : std_logic;

constant CounterMatch : unsigned(11 downto 0) :=
TO_UNSIGNED(BlockSize+1,12);
constant AddrSaturation : unsigned(AccumulatorWidth-1 downto 0) :=
(others => '1');
```

```
constant ThresholdDefault : unsigned(DataWidth-1 downto 0) :=
TO_UNSIGNED(DefaultSigma,DataWidth);
constant MagStatDefault : std_logic_vector(AccumulatorWidth-1 downto
0) := STD_LOGIC_VECTOR(TO_UNSIGNED(MagStatDef,AccumulatorWidth));

signal BlockCounter : unsigned(11 downto 0);
signal LUT_Addr : unsigned(AccumulatorWidth-1 downto 0);

begin

---------- Pipeline Stages ------------------

pipe2: process (CLK, RESET)
    begin
        if (RESET = '1') then
            Pipe3Data <= (others => '0');
            Pipe3Enable <= '0';
        elsif rising_edge(CLK) then
            if (DataEmpty = '1') and (StallPipe = '0') then
                Pipe3Data <= SIGNED(DataIn);
                Pipe3Enable <= '1';
            else
                Pipe3Enable <= '0';
            end if;
        end if;
end process pipe2;

pipe2a: process (CLK, RESET)
    begin
        if (RESET = '1') then
            DataLoad <= '0';
            BlockCounter <= (others => '0');
            LUT_Addr <= (others => '0');
            MagStat <= MagStatDefault;
        elsif rising_edge(CLK) then
            if (DataEmpty = '1') and (StallPipe = '0') then
                if (BlockCounter = CounterMatch) then
                    DataLoad <= '1';
                    MagStat <= STD_LOGIC_VECTOR(LUT_Addr);
                    BlockCounter <= (others => '0');
                    LUT_Addr <= (others => '0');
                else
                    DataLoad <= '0';
                    BlockCounter <= BlockCounter +1;
                    LUT_Addr <= LUT_Addr +
UNSIGNED(abs(SIGNED(DataIn)));
                end if;
            end if;
        end if;
end process pipe2a;
```

```
LUT_AddrBus <= STD_LOGIC_VECTOR(LUT_Addr(AccumulatorWidth-1 downto
AccumulatorWidth-17));


pipe3 : process (CLK, RESET)
    begin
        if (RESET = '1') then
            Pipe4Enable <= '0';
            Pipe4Data <= (others => '0');
        elsif rising_edge(CLK) then
            if (Pipe3Enable = '1') and (StallPipe = '0') then
                Pipe4Data <= Pipe3Data;
                Pipe4Enable <= '1';
            else
                Pipe4Enable <= '0';
            end if;
      end if;
end process pipe3;

pipe4 : process (CLK, RESET)
    begin
        if (RESET = '1') then
            Pipe5Enable <= '0';
            Pipe5Data <= (others => '0');
        elsif rising_edge(CLK) then
            if (Pipe4Enable = '1') and (StallPipe = '0') then
                Pipe5Data <= Pipe4Data;
                Pipe5Enable <= '1';
            else
                Pipe5Enable <= '0';
            end if;
      end if;
end process pipe4;

pipe4a : process (CLK, RESET)
    begin
        if (RESET = '1') then
            Threshold <= ThresholdDefault;
        elsif rising_edge(CLK) then
            if DataLoad = '1' then
                Threshold <= UNSIGNED(LUT_DataBus(DataWidth-1 downto
0));
            end if;
        end if;
end process pipe4a;

pipe5 : process (CLK, RESET)
    begin
        if (RESET = '1') then
            DataOut <= (others => '0');
            DataReady <= '0';
```

```
        elsif rising_edge(CLK) then
            if (Pipe5Enable = '1') and (StallPipe = '0') then
                DataReady <= '1';
                DataOut(1) <= Pipe5Data(DataWidth-1);  --Transfer sign
                if (UNSIGNED(abs(Pipe5Data)) < Threshold) then
                    DataOut(0) <= '0';
                else
                    DataOut(0) <= '1';
                end if;
            else
                DataReady <= '0';
            end if;
      end if;
end process pipe5;

end Behavioral;
```

D.2 1-D Wavelet BAQ Core (mra_core.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity MRA_Core is
    Generic (
    BlockSize        : integer := 702;
       AccumulatorWidth : integer := 27;
    DefaultSigma_A   : integer := 45;
       MagStatDefA  : integer := 0;
    DefaultSigma_D   : integer := 45;
       MagStatDefD      : integer := 0;
    WaveDataInWidth  : integer := 8;
       WaveDataOutWidth : integer := 17;
       WaveCoeffWidth   : integer := 9;
       h0_coeff : integer := 0;
       h1_coeff : integer := 0;
       h2_coeff : integer := 0;
       h3_coeff : integer := 0
    );
    Port (
       UserDataIn : in std_logic_vector(WaveDataInWidth-1 downto 0);
       UserData_RDen : out std_logic;
       UserData_Empty : in std_logic;

       UserDataOut : out std_logic_vector(1 downto 0);
       UserData_WRen : out std_logic;
       UserData_Full : in std_logic;
       MagStat_A : out std_logic_vector(AccumulatorWidth-1 downto 0);
       MagStat_D : out std_logic_vector(AccumulatorWidth-1 downto 0);
       ADturn : out std_logic;

       ZBT1_Addr : out std_logic_vector(16 downto 0);
       ZBT1_Data : in std_logic_vector(31 downto 0);

       ZBT2_Addr : out std_logic_vector(16 downto 0);
       ZBT2_Data : in std_logic_vector(31 downto 0);

       User_CLK : in std_logic;
       CLKdiv2 : in std_logic;
       Reset : in std_logic
    );
end MRA_Core;

architecture Behavioral of MRA_Core is

component Wavelet_Core is
    Generic (
```

```
    DataInWidth : integer;
       DataOutWidth : integer;
       CoeffWidth : integer;
       h0_coeff : integer;
       h1_coeff : integer;
       h2_coeff : integer;
       h3_coeff : integer
    );
    Port (
       DataIn : in std_logic_vector(WaveDataInWidth-1 downto 0);
       Data_Empty : in std_logic;

       DataOutA : out std_logic_vector(WaveDataOutWidth-1 downto 0);
       DataOutD : out std_logic_vector(WaveDataOutWidth-1 downto 0);
       Data_Ready : out std_logic;
       StallPipe : in std_logic;

       CLK : in std_logic;
       CLKdiv2 : in std_logic;
       Reset : in std_logic
    );
end component;

component BAQ_Core is
    Generic (
    BlockSize        : integer;
       AccumulatorWidth : integer;
    DataWidth        : integer;
       MagStatDef       : integer;
    DefaultSigma     : integer
    );
    Port (
       DataIn : in std_logic_vector(WaveDataOutWidth-1 downto 0);
       DataEmpty : in std_logic;

       DataOut : out std_logic_vector(1 downto 0);
       DataReady : out std_logic;
       StallPipe : in std_logic;
       MagStat : out std_logic_vector(AccumulatorWidth-1 downto 0);

       LUT_AddrBus : out std_logic_vector(16 downto 0);
       LUT_DataBus : in std_logic_vector(31 downto 0);

       CLK : in std_logic;
       Reset : in std_logic
    );
end component;


signal WaveletData_Enable : std_logic;
```

```
signal DataOutA, DataOutD : std_logic_vector(WaveDataOutWidth-1
downto 0);
signal BAQ_Enable : std_logic;
signal CompDataOutA, CompDataOutD : std_logic_vector(1 downto 0);
signal BAQ_A_Ready, BAQ_D_Ready : std_logic;
signal notCLKdiv2 : std_logic;
signal StallPipe : std_logic;

Begin

StallPipe <= UserData_Full;
notCLKdiv2 <= not CLKdiv2;

---------- Component Instantiation ====================
WCORE: Wavelet_Core
    Generic Map(
    DataInWidth => WaveDataInWidth,
       DataOutWidth => WaveDataOutWidth,
       CoeffWidth => WaveCoeffWidth,
       h0_coeff => h0_coeff,
       h1_coeff => h1_coeff,
       h2_coeff => h2_coeff,
       h3_coeff => h3_coeff
    )
    Port Map(
       DataIn => UserDataIn,
       Data_Empty => WaveletData_Enable,
       DataOutA => DataOutA,
       DataOutD => DataOutD,
       Data_Ready => BAQ_Enable,
       StallPipe => StallPipe,
       CLK => User_CLK,
       CLKdiv2 => CLKdiv2,
       Reset => Reset
    );

BAQ_A : BAQ_Core
    Generic Map(
    BlockSize => BlockSize,
       AccumulatorWidth => AccumulatorWidth,
    DataWidth => WaveDataOutWidth,
       MagStatDef => MagStatDefA,
    DefaultSigma => DefaultSigma_A
    )
    Port Map(
       DataIn => DataOutA,
       DataEmpty => BAQ_Enable,
       DataOut => CompDataOutA,
       DataReady => BAQ_A_Ready,
       StallPipe => StallPipe,
       MagStat => MagStat_A,
```

```
        LUT_AddrBus => ZBT1_Addr,
        LUT_DataBus => ZBT1_Data,
        CLK => CLKdiv2,
        Reset => Reset
     );

BAQ_D : BAQ_Core
    Generic Map(
    BlockSize => BlockSize,
        AccumulatorWidth => AccumulatorWidth,
    DataWidth => WaveDataOutWidth,
        MagStatDef => MagStatDefD,
    DefaultSigma => DefaultSigma_D
    )
    Port Map(
        DataIn => DataOutD,
        DataEmpty => BAQ_Enable,
        DataOut => CompDataOutD,
        DataReady => BAQ_D_Ready,
        StallPipe => StallPipe,
        MagStat => MagStat_D,
        LUT_AddrBus => ZBT2_Addr,
        LUT_DataBus => ZBT2_Data,
        CLK => CLKdiv2,
        Reset => Reset
     );

----------- Front End FSM Framer --------------------
FSM1: process (User_CLK, RESET)
    begin
        if (RESET = '1') then
            UserData_RDen <= '0';
            WaveletData_Enable <= '0';
        elsif falling_edge(User_CLK) then
            if (UserData_Empty = '0') and (StallPipe = '0') then
                UserData_RDen <= '1';
                WaveletData_Enable <= '1';
            else
                UserData_RDen <= '0';
                WaveletData_Enable <= '0';
            end if;
      end if;
end process FSM1;

----------- Back End FSM Framer --------------------
BFSM1: process (User_CLK, RESET)
    variable ADturn_var : std_logic;
    begin
        if (RESET = '1') then
            UserData_WRen <= '0';
            UserDataOut <= (others => '0');
```

```
            ADturn_var := '0';
        elsif rising_edge(User_CLK) then
            if (UserData_Full = '0') then
                if (ADturn_var = '0') and (BAQ_A_Ready = '1') then --
Read A number
                    UserData_WRen <= '1';
                    UserDataOut <= CompDataOutA;
                    ADturn_var := '1';
                elsif (ADturn_var = '1') and (BAQ_D_Ready = '1') then -
- Read D number
                    UserData_WRen <= '1';
                    UserDataOut <= CompDataOutD;
                    ADturn_var := '0';
                else
                    UserData_WRen <= '0';
                end if;
            else
                UserData_WRen <= '0';
            end if;
        end if;
    ADturn <= ADturn_var;
end process BFSM1;

end Behavioral;
```

D.3 Wavelet Block (wavelet_core.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity Wavelet_Core is
    Generic (
    DataInWidth : integer := 8;
       DataOutWidth : integer := 17;
       CoeffWidth : integer := 9;
       h0_coeff : integer := 0;
       h1_coeff : integer := 0;
       h2_coeff : integer := 0;
       h3_coeff : integer := 0
    );
    Port (
       DataIn : in std_logic_vector(DataInWidth-1 downto 0);
       Data_Empty : in std_logic;

       DataOutA : out std_logic_vector(DataOutWidth-1 downto 0);
       DataOutD : out std_logic_vector(DataOutWidth-1 downto 0);
       Data_Ready : out std_logic;
       StallPipe : in std_logic;

       CLK : in std_logic;
       CLKdiv2 : in std_logic;
       Reset : in std_logic
    );
end Wavelet_Core;

architecture Behavioral of Wavelet_Core is

component CoeffMultiplier_Core is
    Generic (
       DataInWidth : integer := 8;
       DataOutWidth : integer := 17;
       CoeffWidth : integer := 9;
       h0A_coeff : integer := 0;
       h1A_coeff : integer := 0;
       h0D_coeff : integer := 0;
       h1D_coeff : integer := 0
   );
    Port (
       DataIn : in signed(DataInWidth-1 downto 0);
       Data_Empty : in std_logic;
       DataOutA0 : out signed(DataOutWidth-1 downto 0);
       DataOutD0 : out signed(DataOutWidth-1 downto 0);
       DataOutA1 : out signed(DataOutWidth-1 downto 0);
       DataOutD1 : out signed(DataOutWidth-1 downto 0);
```

```
        Data_Ready : out std_logic;

        StallPipe : in std_logic;

        CLK : in std_logic;
        Reset : in std_logic
    );
end component;

signal OddDataIn, EvenDataIn : signed(DataInWidth-1 downto 0);
signal OddDataOutA0, OddDataOutD0, OddDataOutA1, OddDataOutD1 :
signed(DataOutWidth-1 downto 0);
signal EvenDataOutA0, EvenDataOutD0, EvenDataOutA1, EvenDataOutD1 :
signed(DataOutWidth-1 downto 0);
signal OddData_Ready, EvenData_Ready : std_logic;

signal OddData_Empty, EvenData_Empty : std_logic;

signal notCLKdiv2 : std_logic;

begin

notCLKdiv2 <= not CLKdiv2;

---------- ComponentInstantiation ===================


odd_mult: CoeffMultiplier_Core
Generic Map(
        DataInWidth => DataInWidth,
    DataOutWidth => DataOutWidth,
        CoeffWidth => CoeffWidth,
    h0A_coeff => h0_coeff,
    h1A_coeff => h2_coeff,
    h0D_coeff => -h3_coeff,
    h1D_coeff => -h1_coeff
)
Port Map(
    DataIn => OddDataIn,
    Data_Empty => OddData_Empty,
    DataOutA0 => OddDataOutA0,
    DataOutD0 => OddDataOutD0,
    DataOutA1 => OddDataOutA1,
    DataOutD1 => OddDataOutD1,
    Data_Ready => OddData_Ready,
    StallPipe => StallPipe,
    CLK => notCLKdiv2,
    Reset => Reset
);

even_mult: CoeffMultiplier_Core
```

```
Generic Map(
        DataInWidth => DataInWidth,
    DataOutWidth => DataOutWidth,
        CoeffWidth => CoeffWidth,
    h0A_coeff => h1_coeff,
    h1A_coeff => h3_coeff,
    h0D_coeff => h2_coeff,
    h1D_coeff => h0_coeff
)
Port Map(
    DataIn => EvenDataIn,
    Data_Empty => EvenData_Empty,
    DataOutA0 => EvenDataOutA0,
    DataOutD0 => EvenDataOutD0,
    DataOutA1 => EvenDataOutA1,
    DataOutD1 => EvenDataOutD1,
    Data_Ready => EvenData_Ready,
    StallPipe => StallPipe,
    CLK => notCLKdiv2,
    Reset => Reset
 );

----------- Front End FSM --------------------
FSM1: process (CLK, RESET)
    variable EvenData_DataInStall : signed(DataInWidth-1 downto 0);
    variable EvenOdd : std_logic;
    variable FirstOdd : signed(DataInWidth-1 downto 0);
    variable FirstEven : signed(DataInWidth-1 downto 0);
    variable FirstOddLoad : std_logic;
    variable FirstEvenLoad : std_logic;
    variable onewait : std_logic;
    begin
        if (RESET = '1') then
            OddData_Empty <= '0';
            EvenData_Empty <= '0';
            OddDataIn <= (others => '0');
            EvenDataIn <= (others => '0');
            EvenData_DataInStall := (others => '0');
            EvenOdd := '0';
            FirstOddLoad := '1';
            FirstEvenLoad := '1';
            FirstOdd := (others => '0');
            FirstEven := (others => '0');
            onewait := '0';
        elsif rising_edge(CLK) then
            if (Data_Empty = '1') and (StallPipe = '0') then
                if (EvenOdd = '0') then -- even number
                    EvenData_Empty <= '1';
                    EvenDataIn <= EvenData_DataInStall;
                    EvenData_DataInStall := SIGNED(UNSIGNED(DataIn) -
128);
```

```
                    EvenOdd := '1';
                    if (FirstEvenLoad = '1') then
                        FirstEven := SIGNED(UNSIGNED(DataIn) - 128);
                        FirstEvenLoad := '0';
                    end if;
                else
                    OddData_Empty <= '1';
                    OddDataIn <= SIGNED(UNSIGNED(DataIn) - 128);
                    EvenDataIn <= EvenData_DataInStall;
                    EvenOdd := '0';
                    if (FirstOddLoad = '1') then
                        FirstOdd := SIGNED(UNSIGNED(DataIn) - 128);
                        FirstOddLoad := '0';
                    end if;
                end if;
            else
                if (EvenOdd = '0') and (FirstEvenLoad = '0') then
                    EvenData_Empty <= '1';
                    EvenDataIn <= EvenData_DataInStall;
                    EvenData_DataInStall := FirstEven;
                    EvenOdd := '1';
                    FirstEvenLoad := '1';
                elsif (EvenOdd = '1') and (FirstOddLoad = '0') then
                    OddData_Empty <= '1';
                    OddDataIn <= FirstOdd;
                    EvenDataIn <= EvenData_DataInStall;
                    EvenOdd := '0';
                    FirstOddLoad := '1';
                else
                    if (onewait = '0') then
                        onewait := '1';
                    else
                        EvenData_Empty <= '0';
                        OddData_Empty <= '0';
                    end if;
                end if;

            end if;
        end if;
end process FSM1;

--========== Back End Framer ====================

B_DOREG: process (CLKdiv2, RESET)
begin
    if (RESET = '1') then
        DataOutA <= (others => '0');
        DataOutD <= (others => '0');
        Data_Ready <= '0';
    elsif rising_edge(CLKdiv2) then
        if (OddData_Ready = '1') and (EvenData_Ready = '1') then
```

```
            DataOutA <= std_logic_vector(EvenDataOutA0 + EvenDataOutA1
+ OddDataOutA0 + OddDataOutA1);
            DataOutD <= std_logic_vector(EvenDataOutD0 + EvenDataOutD1
+ OddDataOutD0 + OddDataOutD1);
            Data_Ready <= '1';
        else
            Data_Ready <= '0';
        end if;
    end if;
end process B_DOREG;

end Behavioral;
```

D.4 Coefficient Multiplier Block (coeff_multiplier_core.vhd)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity CoeffMultiplier_Core is
    Generic (
        DataInWidth : integer := 8;
        DataOutWidth : integer := 17;
        CoeffWidth : integer := 9;
        h0A_coeff : integer := 0;
        h1A_coeff : integer := 0;
        h0D_coeff : integer := 0;
        h1D_coeff : integer := 0
    );
    Port (
        DataIn : in signed(DataInWidth-1 downto 0);
        Data_Empty : in std_logic;
        DataOutA0 : out signed(DataOutWidth-1 downto 0);
        DataOutD0 : out signed(DataOutWidth-1 downto 0);
        DataOutA1 : out signed(DataOutWidth-1 downto 0);
        DataOutD1 : out signed(DataOutWidth-1 downto 0);
        Data_Ready : out std_logic;

        StallPipe : in std_logic;

        CLK : in std_logic;
        Reset : in std_logic
    );
end CoeffMultiplier_Core;

architecture Behavioral of CoeffMultiplier_Core is


signal Pipe2Enable, Pipe3Enable : std_logic;
signal Pipe2Data, Pipe3Data : signed(DataInWidth-1 downto 0);

begin

--========== Pipeline Stages ==================

pipe1 : process (CLK, RESET) -- Read In Data
   begin
       if (RESET = '1') then
           Pipe2Enable <= '0';
           Pipe2Data <= (others => '0');
       elsif rising_edge(CLK) then
           if (Data_Empty = '1') and (StallPipe = '0') then
               Pipe2Data <= DataIn;
```

```
                Pipe2Enable <= '1';
            else
                Pipe2Enable <= '0';
            end if;
        end if;
end process pipe1;

pipe2 : process (CLK, RESET)
    begin
        if (RESET = '1') then
            Pipe3Enable <= '0';
            Pipe3Data <= (others => '0');
            DataOutA0 <= (others => '0');
            DataOutD0 <= (others => '0');
        elsif rising_edge(CLK) then
            if (Pipe2Enable = '1') and (StallPipe = '0') then
                Pipe3Data <= Pipe2Data;
                Pipe3Enable <= '1';
                DataOutA0 <=
TO_SIGNED(h0A_coeff,CoeffWidth)*Pipe2Data;
                DataOutD0 <=
TO_SIGNED(h0D_coeff,CoeffWidth)*Pipe2Data;
            else
                Pipe3Enable <= '0';
            end if;
        end if;
end process pipe2;

pipe3 : process (CLK, RESET)
    begin
        if (RESET = '1') then
            Data_Ready <= '0';
            DataOutA1 <= (others => '0');
            DataOutD1 <= (others => '0');
        elsif rising_edge(CLK) then
            if (Pipe2Enable = '1') and (Pipe3Enable = '1') and
(StallPipe = '0') then
                Data_Ready <= '1';
                DataOutA1 <=
TO_SIGNED(h1A_coeff,CoeffWidth)*Pipe3Data;
                DataOutD1 <=
TO_SIGNED(h1D_coeff,CoeffWidth)*Pipe3Data;
            else
                Data_Ready <= '0';
            end if;
        end if;
end process pipe3;

end Behavioral;
```

APPENDIX E
RESULTS OF MATLAB WAVELET-BAQ

E.1 Statistics of MRA Coefficients

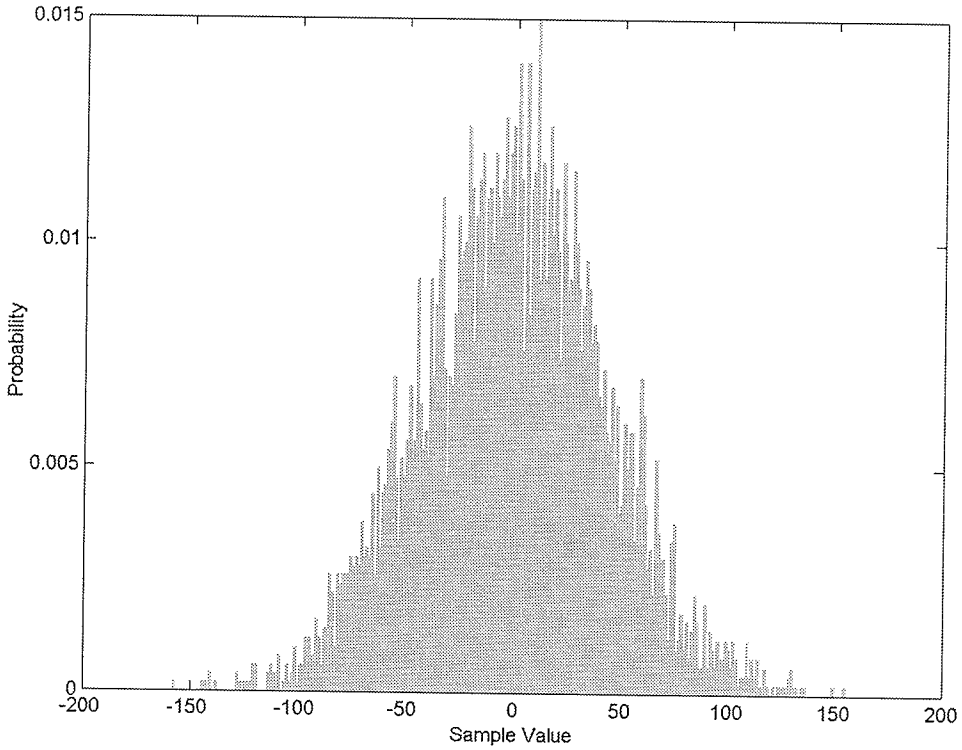


Fig. E.1. Histogram of A_1 from first 10000 samples of E1-22089PS test set.

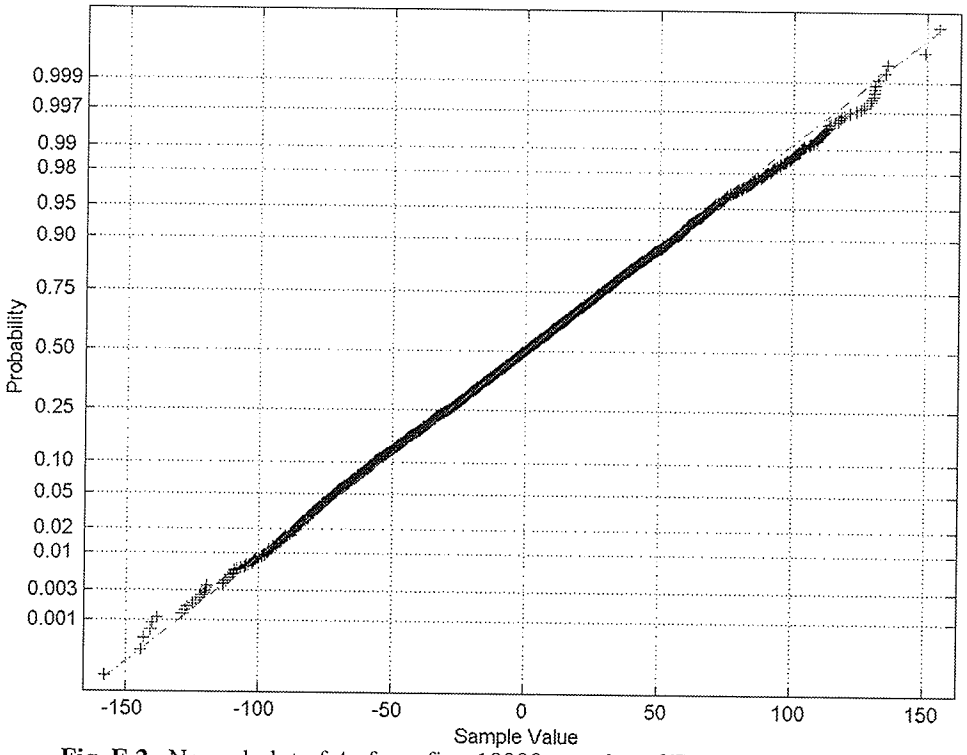


Fig. E.2. Normal plot of A_1 from first 10000 samples of E1-22089PS test set.

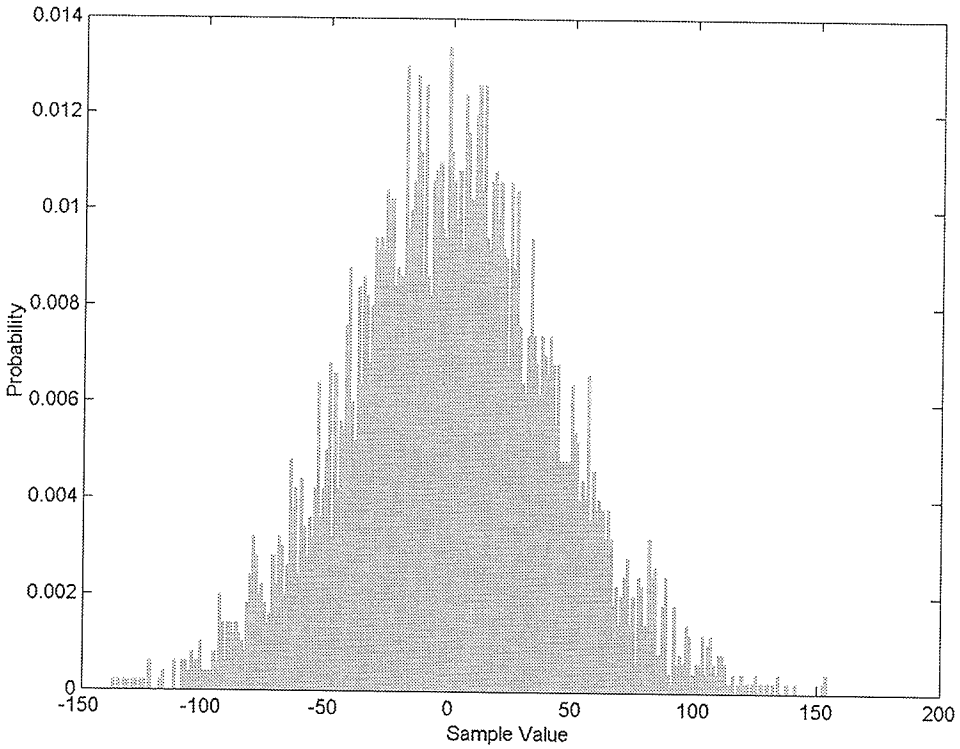


Fig. E.3. Histogram of D_1 from first 10000 samples of E1-22089PS test set.

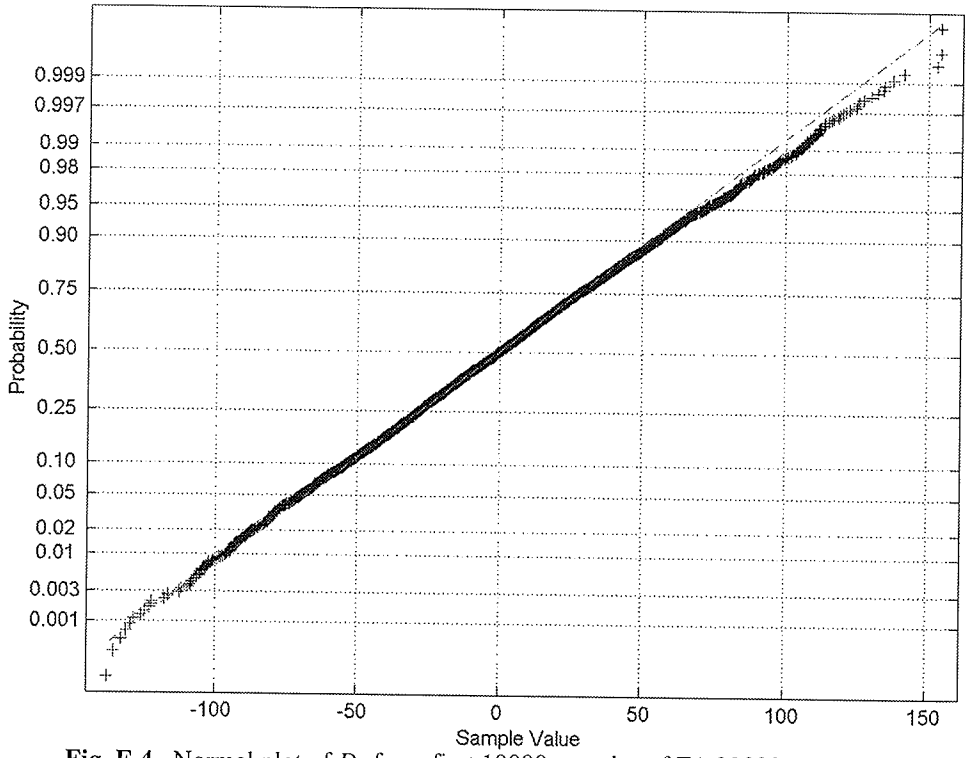


Fig. E.4. Normal plot of D_1 from first 10000 samples of E1-22089PS test set.

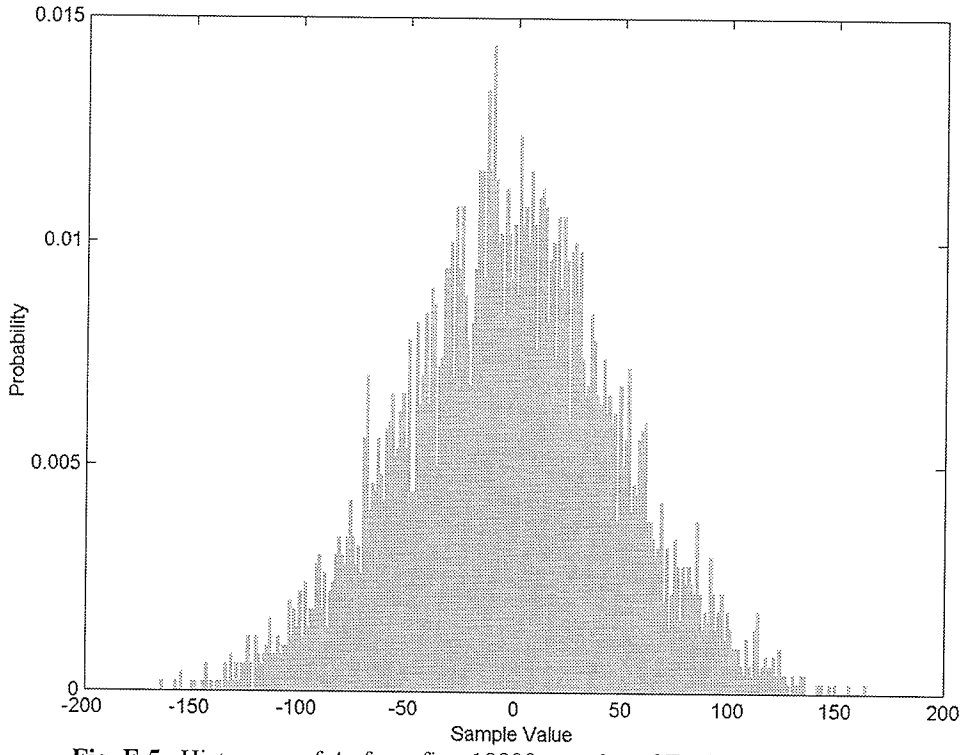


Fig. E.5. Histogram of A_1 from first 10000 samples of E1-25224PS test set.

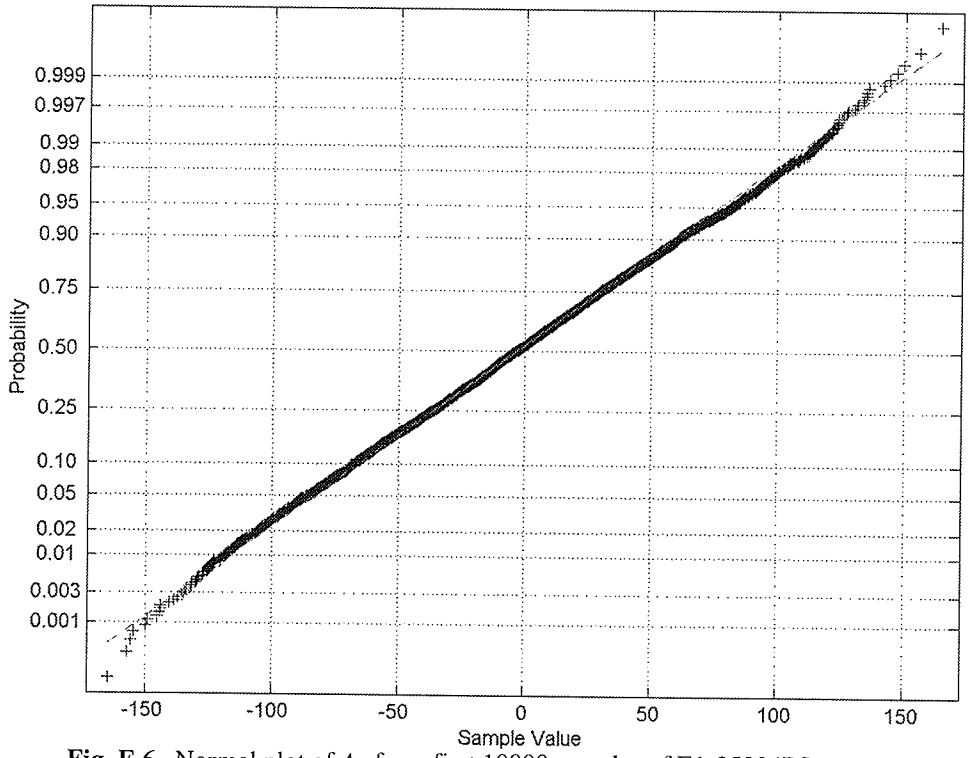


Fig. E.6. Normal plot of A_1 from first 10000 samples of E1-25224PS test set.

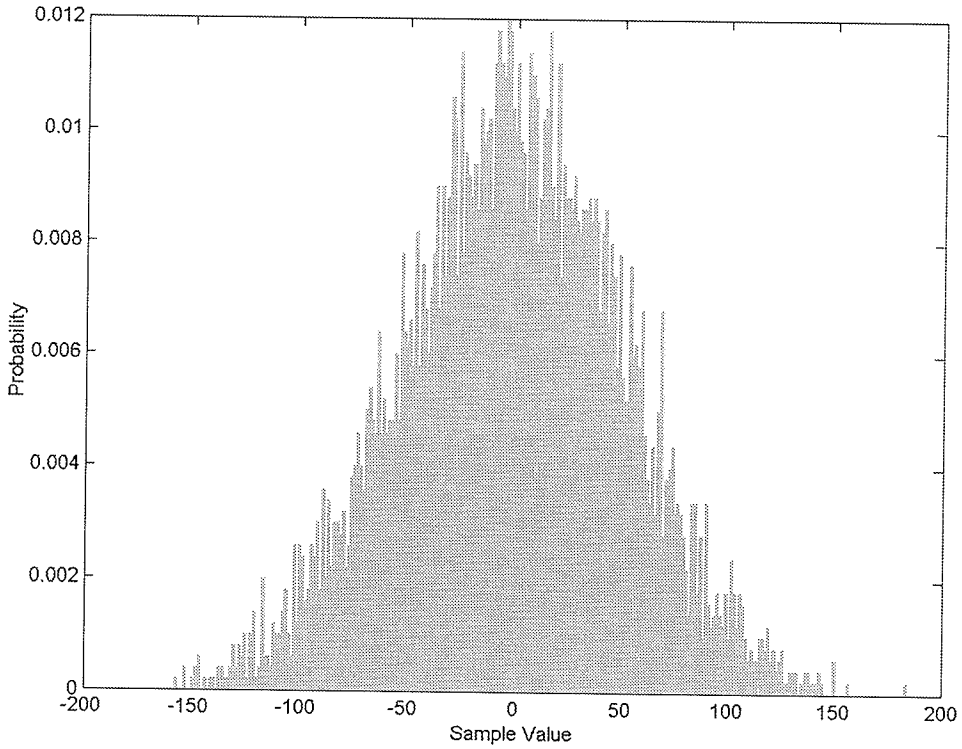


Fig. E.7. Histogram of D_1 from first 10000 samples of E1-25224PS test set.

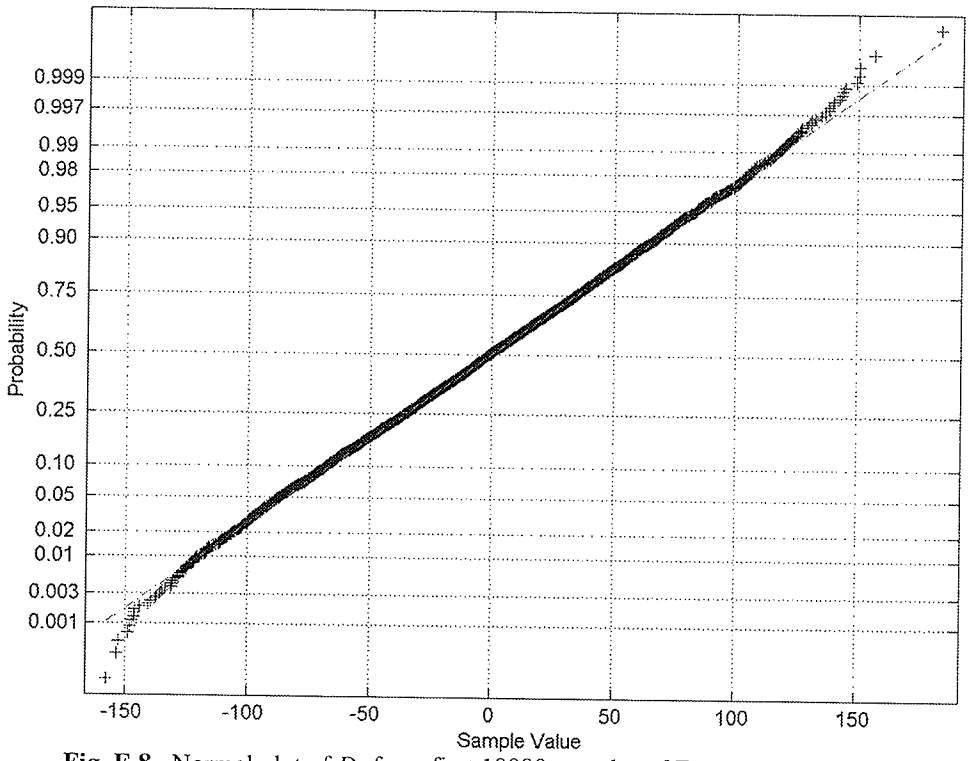


Fig. E.8. Normal plot of D_1 from first 10000 samples of E1-25224PS test set.

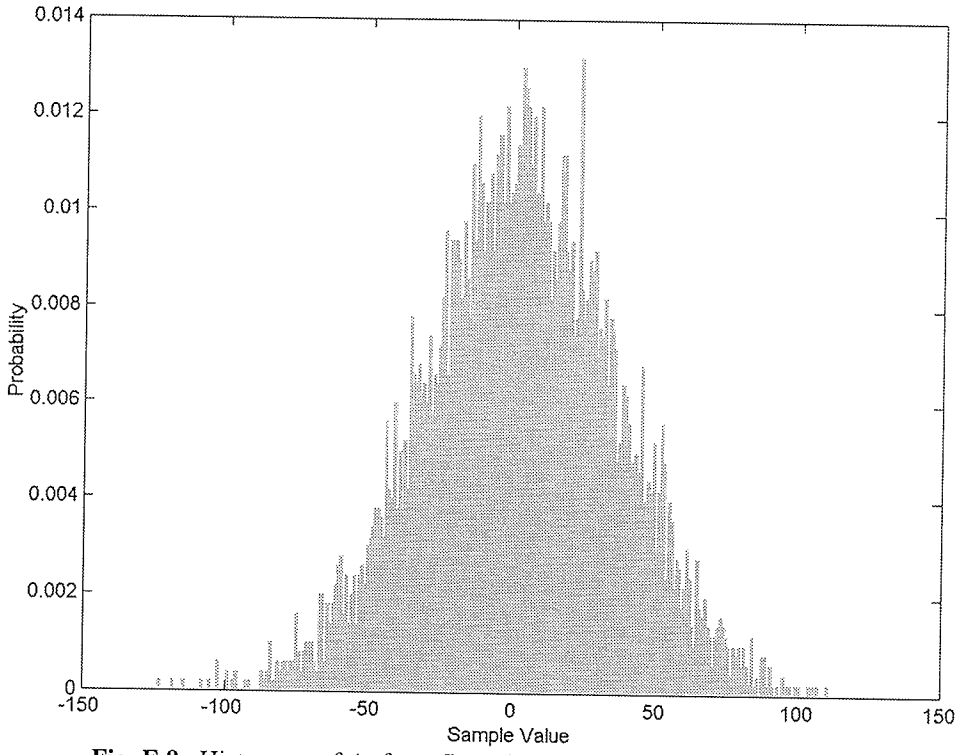


Fig. E.9. Histogram of A_1 from first 10000 samples of E2-5551 test set.

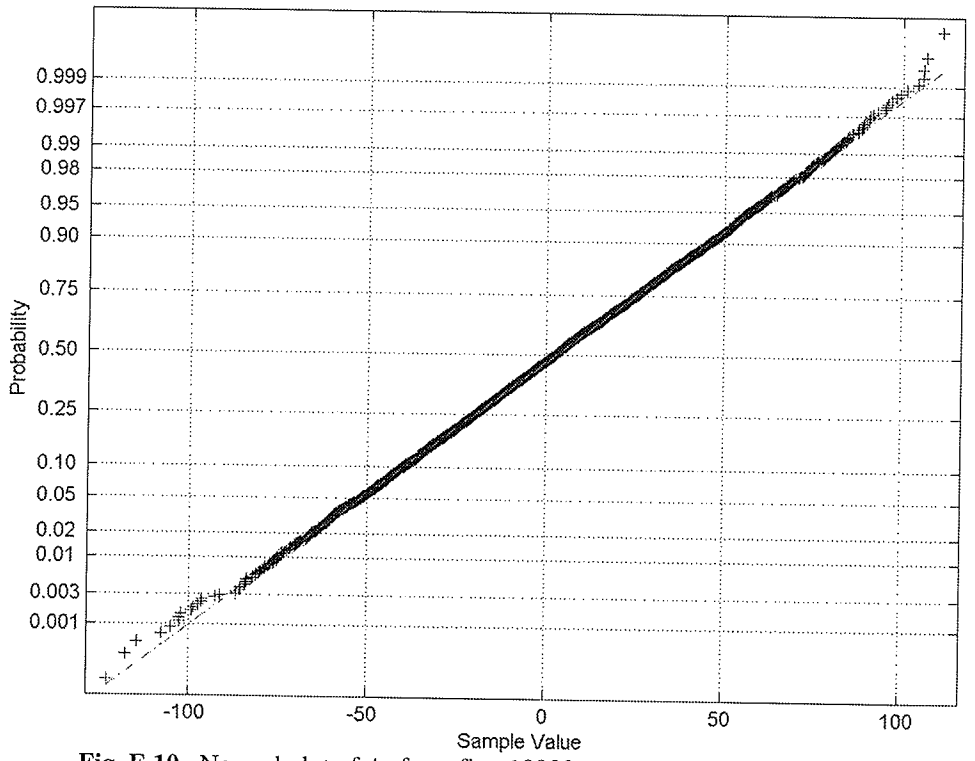


Fig. E.10. Normal plot of A_1 from first 10000 samples of E2-5551 test set.

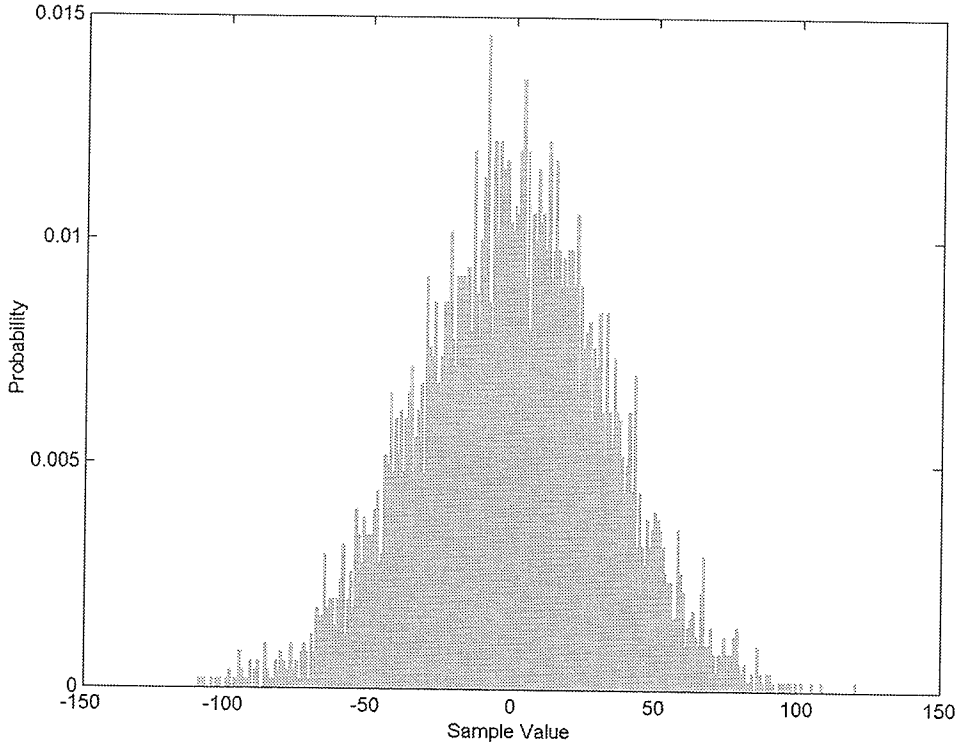


Fig. E.11. Histogram of D_1 from first 10000 samples of E2-5551 test set.

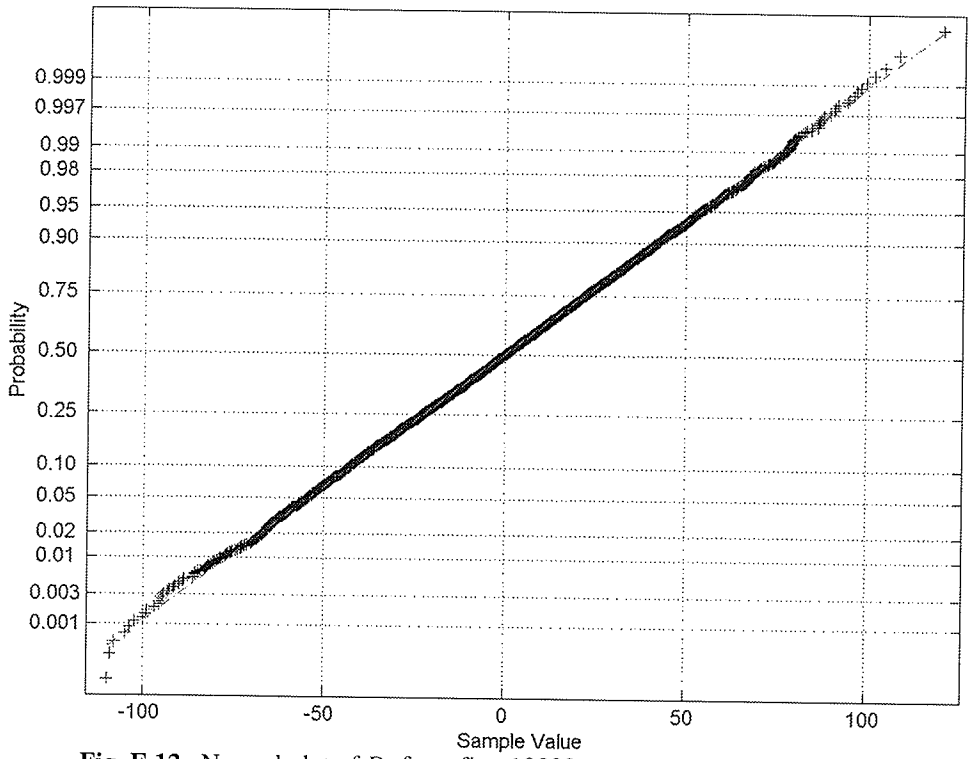


Fig. E.12. Normal plot of D_1 from first 10000 samples of E2-5551 test set.

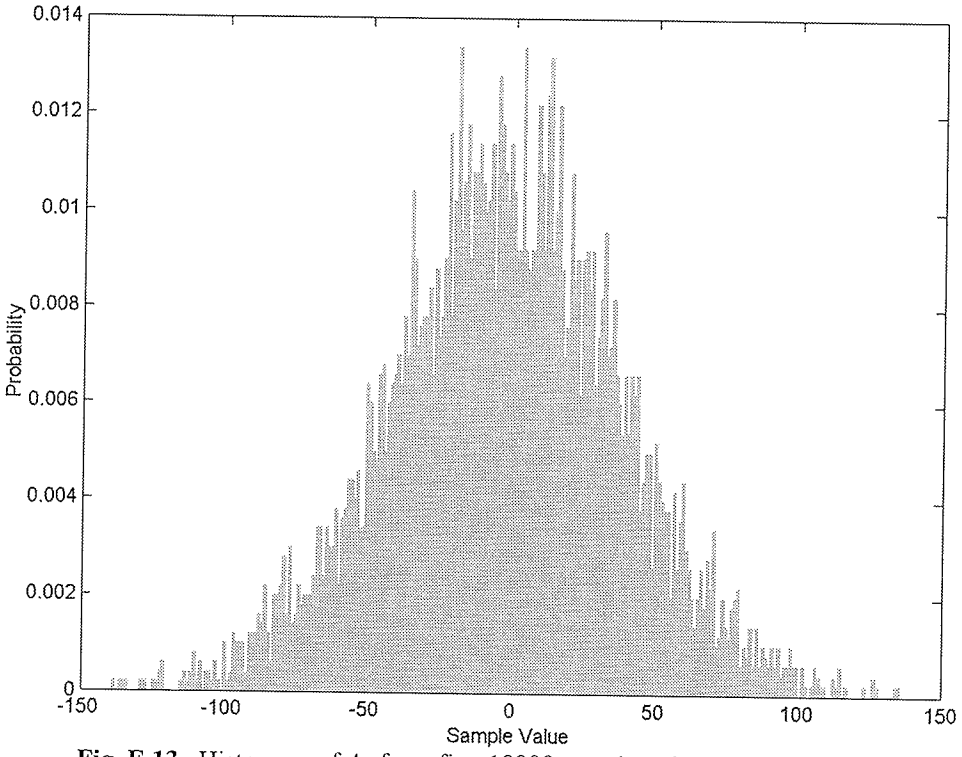


Fig. E.13. Histogram of A_1 from first 10000 samples of R1-24576 test set.

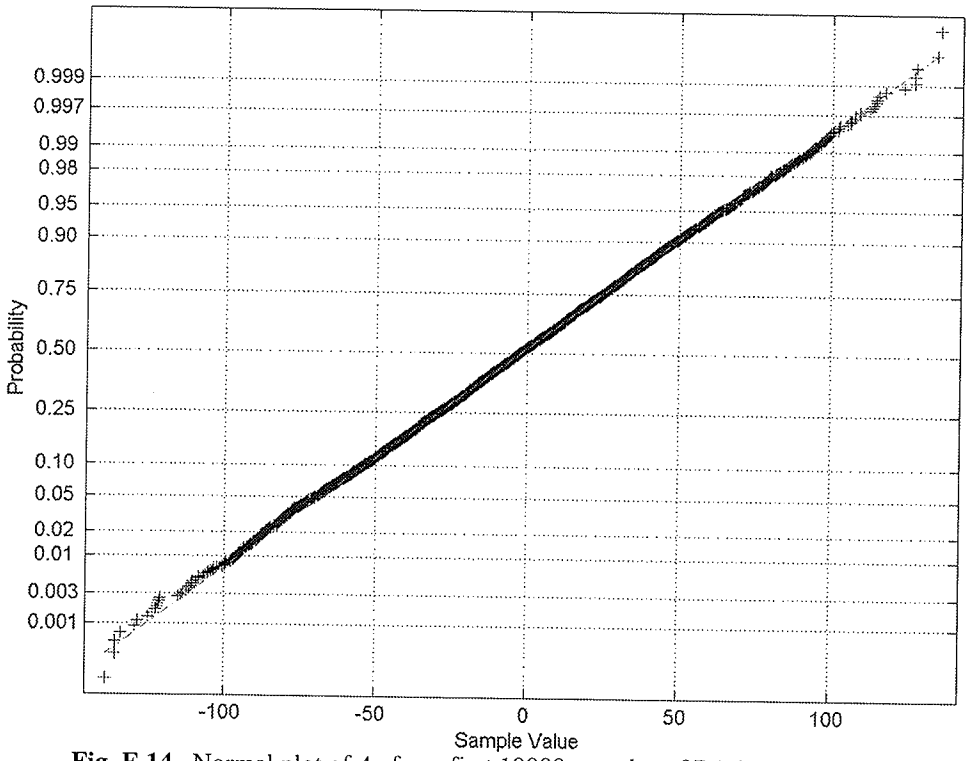


Fig. E.14. Normal plot of A_1 from first 10000 samples of R1-24576 test set.

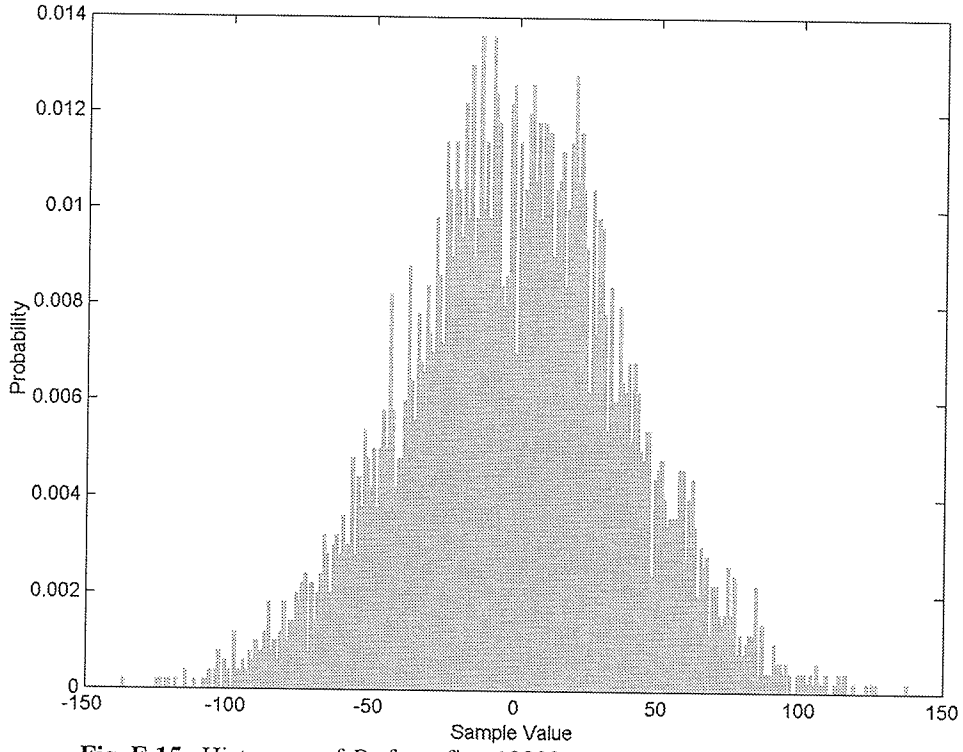


Fig. E.15. Histogram of D_1 from first 10000 samples of R1-24576 test set.

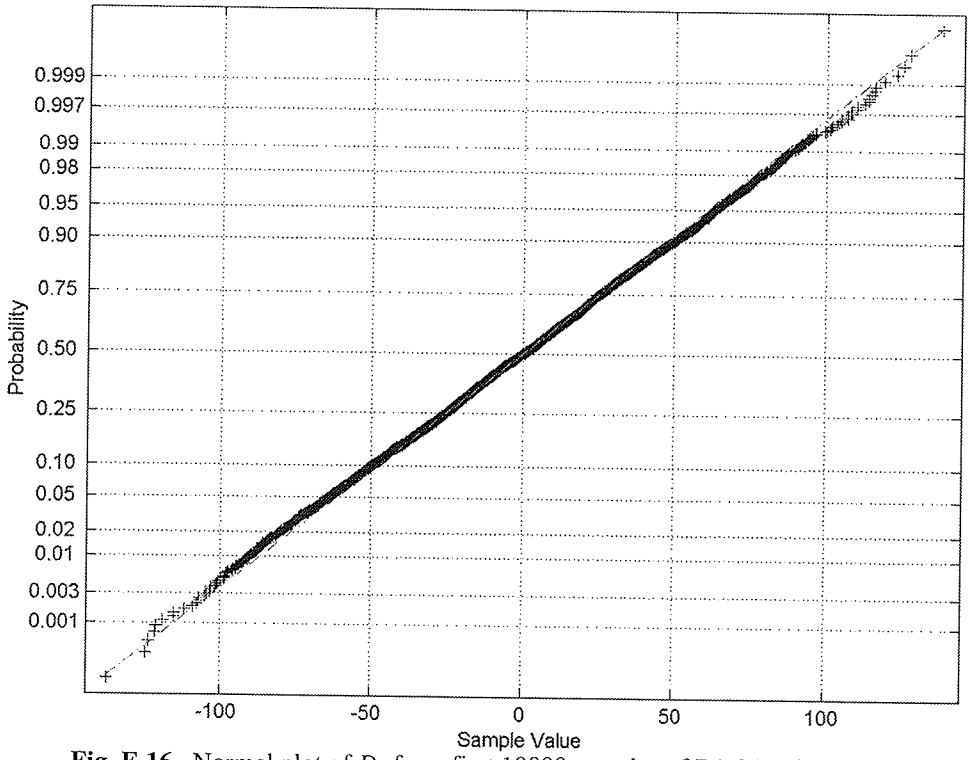


Fig. E.16. Normal plot of D_1 from first 10000 samples of R1-24576 test set.

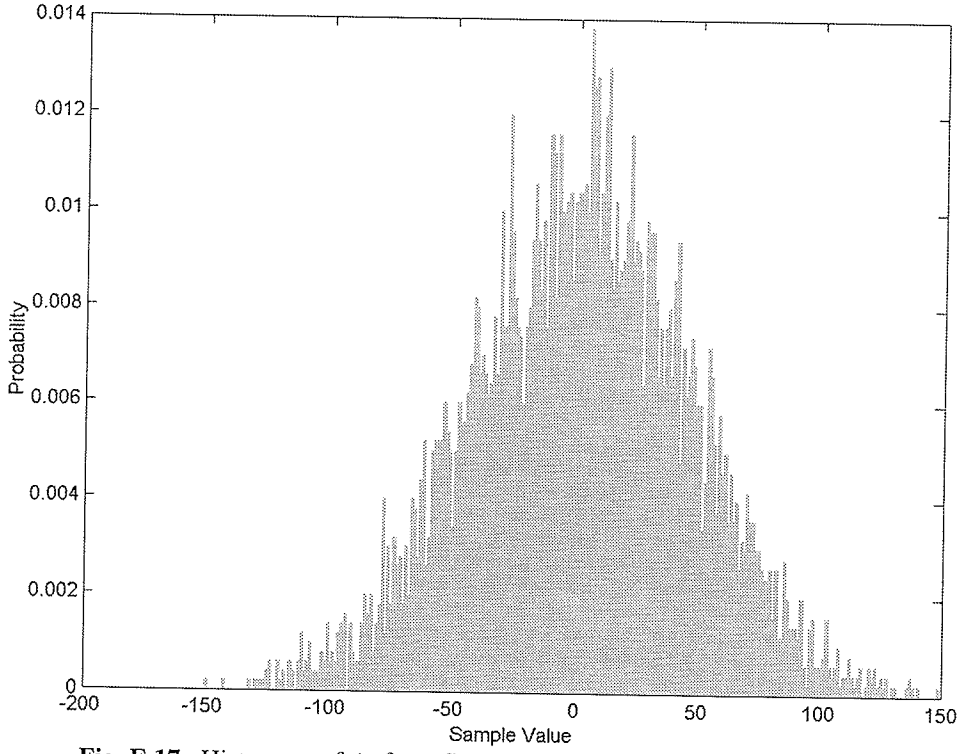


Fig. E.17. Histogram of A_1 from first 10000 samples of R1-24919 test set.

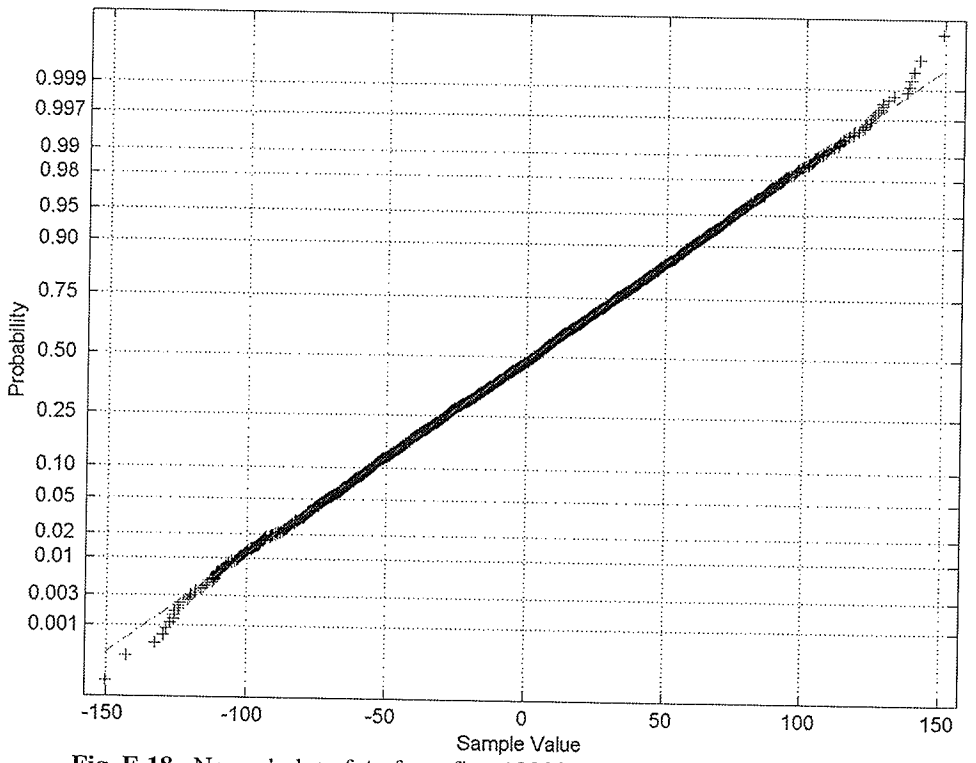


Fig. E.18. Normal plot of A_1 from first 10000 samples of R1-24919 test set.

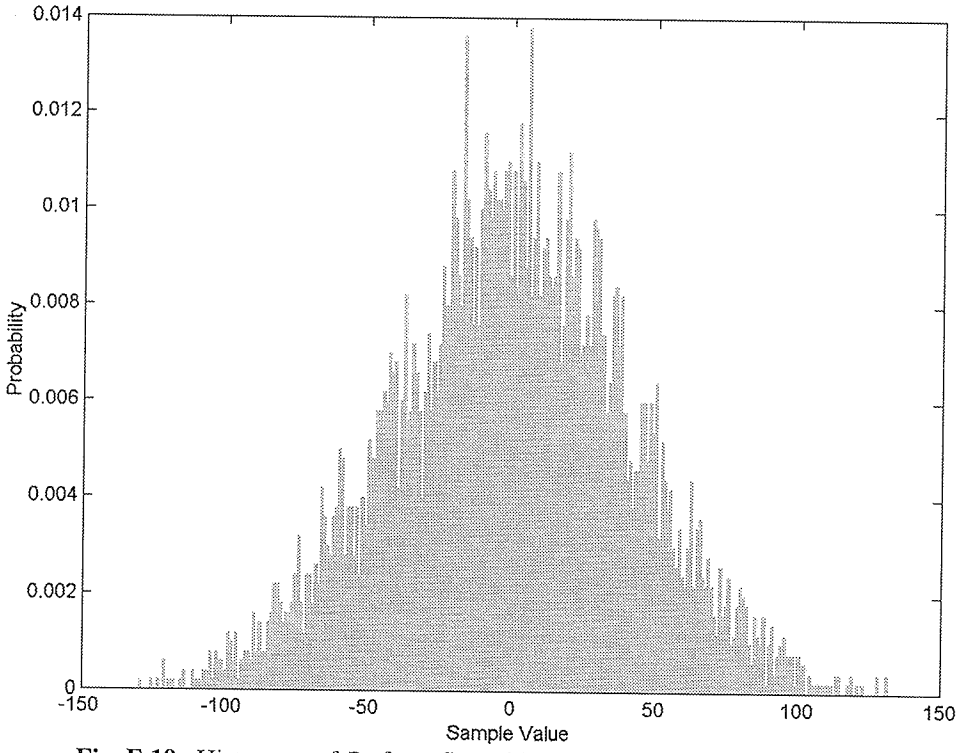


Fig. E.19. Histogram of D_1 from first 10000 samples of R1-24919 test set.

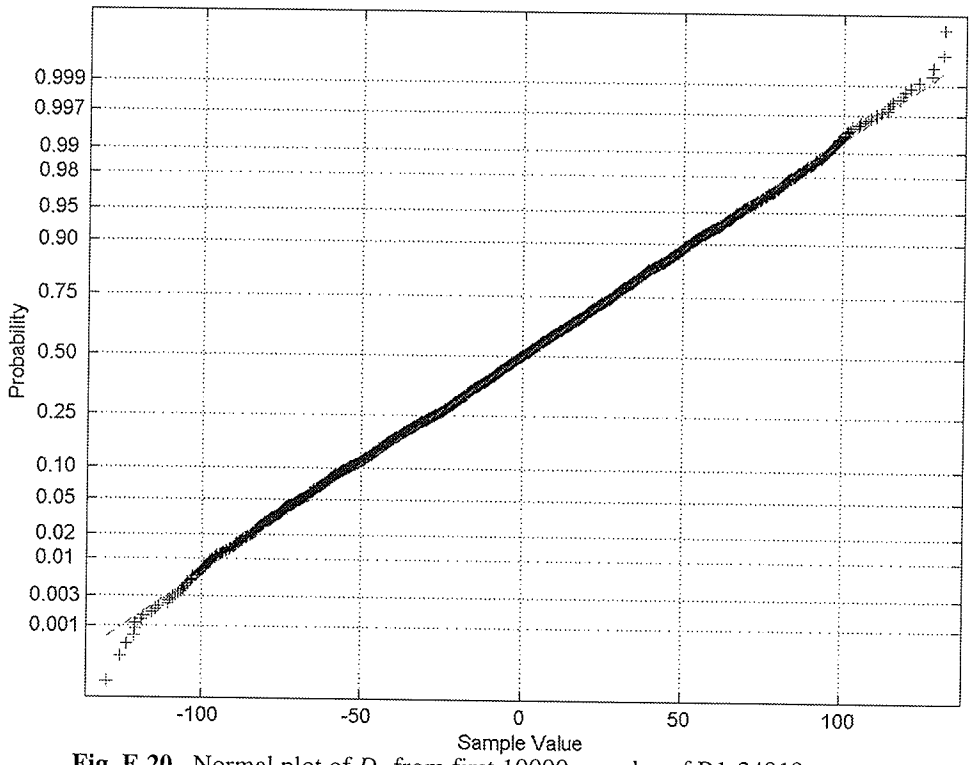


Fig. E.20. Normal plot of D_1 from first 10000 samples of R1-24919 test set.

E.2 Bit Allocation Results

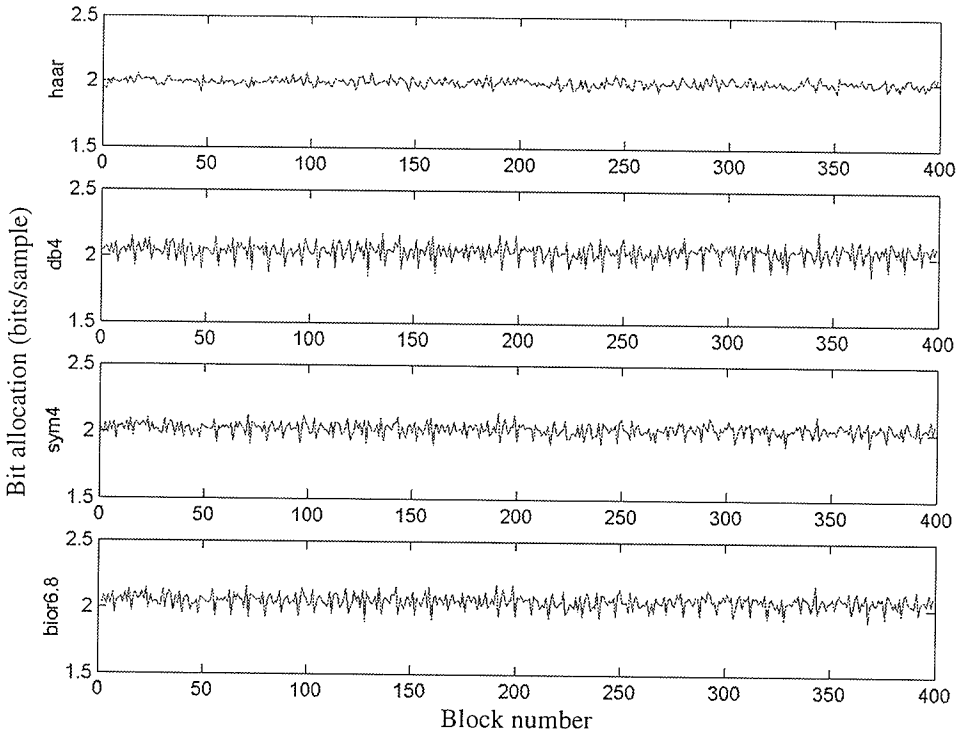


Fig. E.21. A_1 bit allocation from the E1-22089PS test set with Haar, db4, sym4, and bior6.8 wavelets.

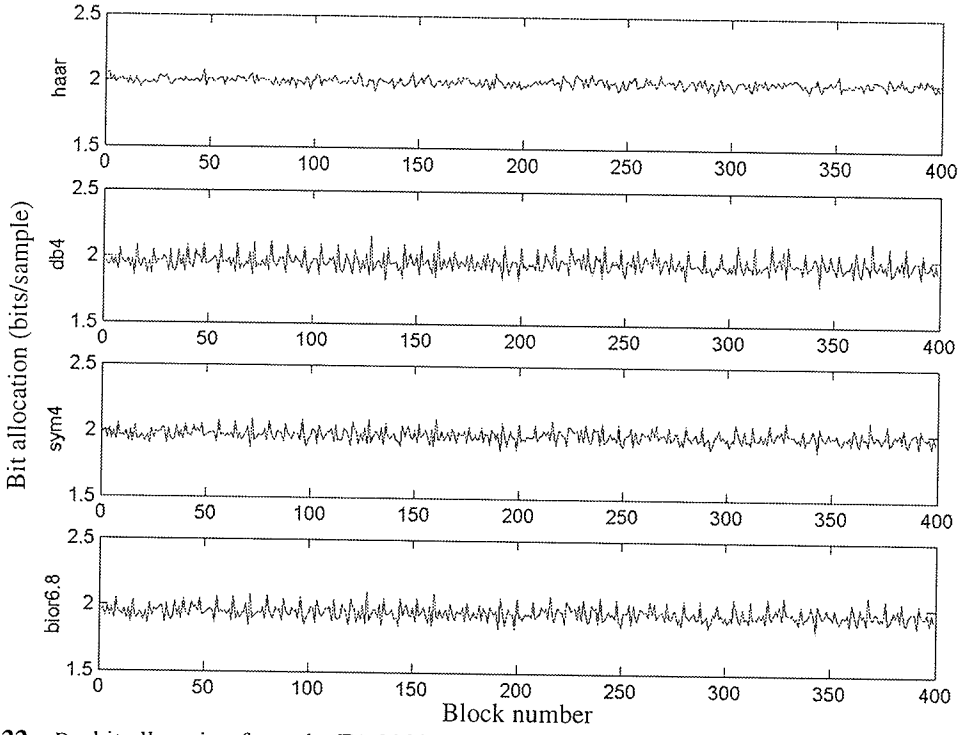


Fig. E.22. D_1 bit allocation from the E1-22089PS test set with Haar, db4, sym4, and bior6.8 wavelets.

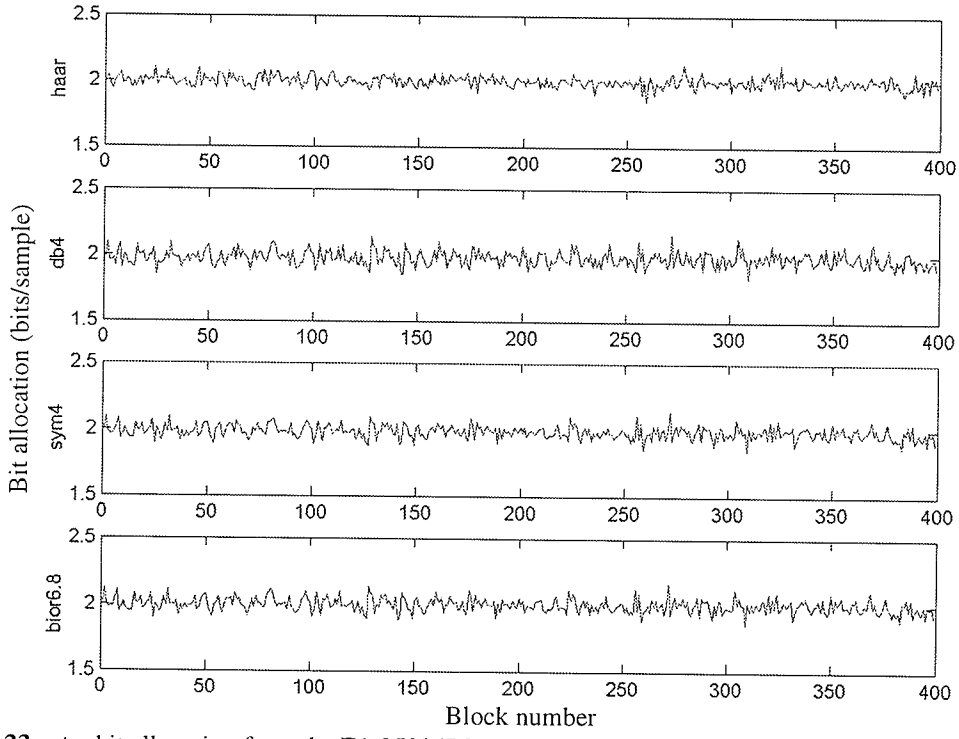


Fig. E.23. A_1 bit allocation from the E1-25224PS test set with Haar, db4, sym4, and bior6.8 wavelets.

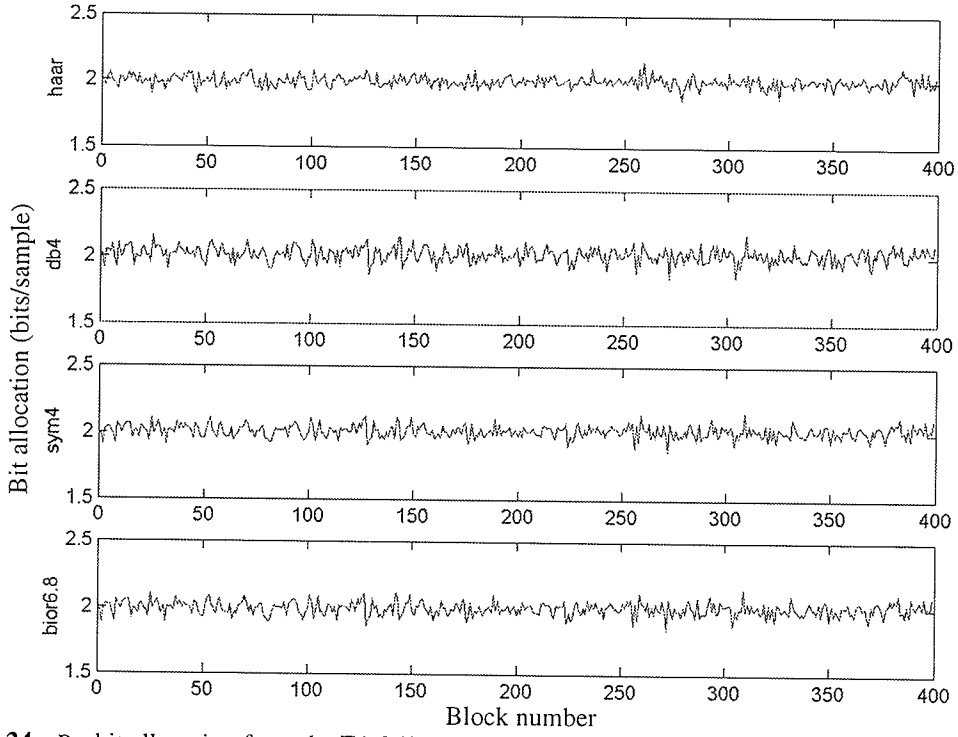


Fig. E.24. D_1 bit allocation from the E1-25224PS test set with Haar, db4, sym4, and bior6.8 wavelets.

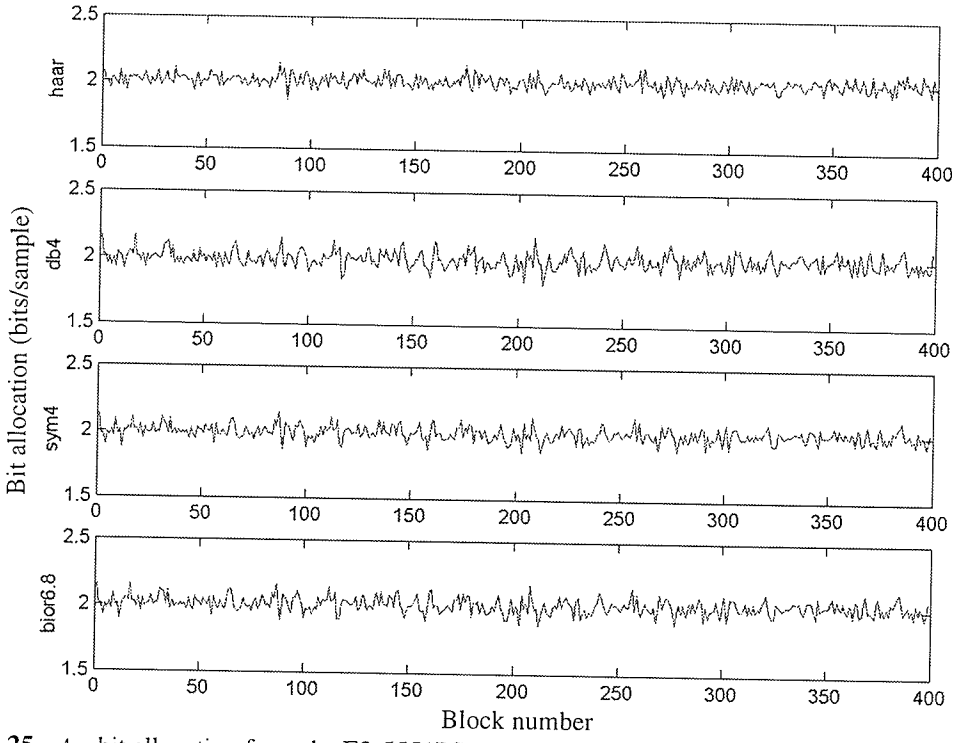


Fig. E.25. A_1 bit allocation from the E2-5551PS test set with Haar, db4, sym4, and bior6.8 wavelets.

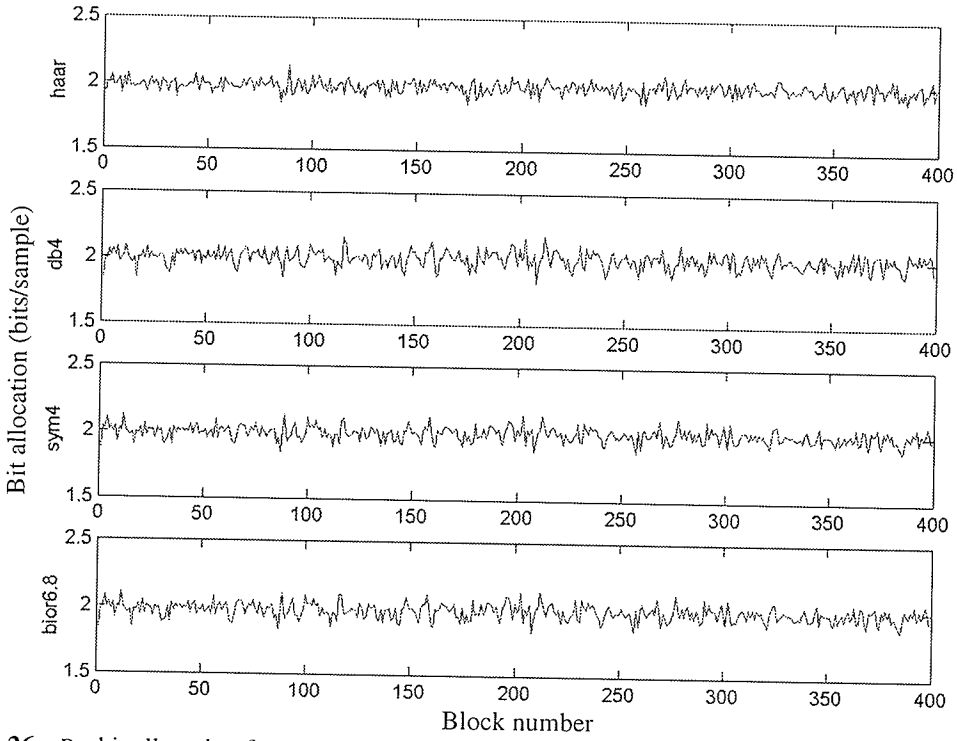


Fig. E.26. D_1 bit allocation from the E2-5551PS test set with Haar, db4, sym4, and bior6.8 wavelets.

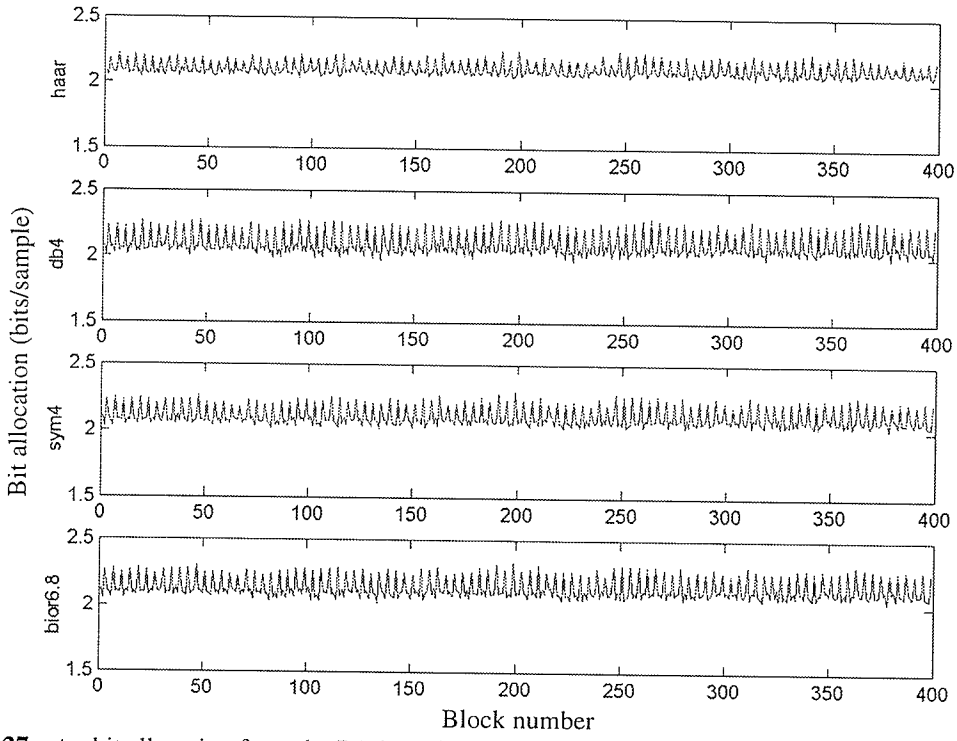


Fig. E.27. A_1 bit allocation from the R1-24576PS test set with Haar, db4, sym4, and bior6.8 wavelets.

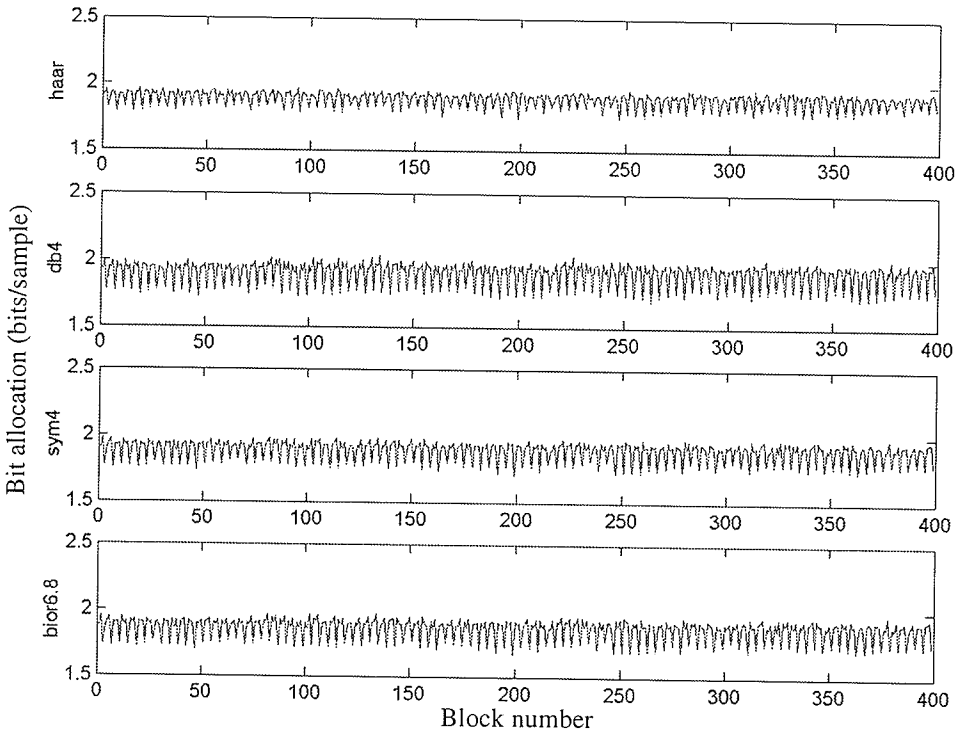


Fig. E.28. D_1 bit allocation from the R1-24576PS test set with Haar, db4, sym4, and bior6.8 wavelets.

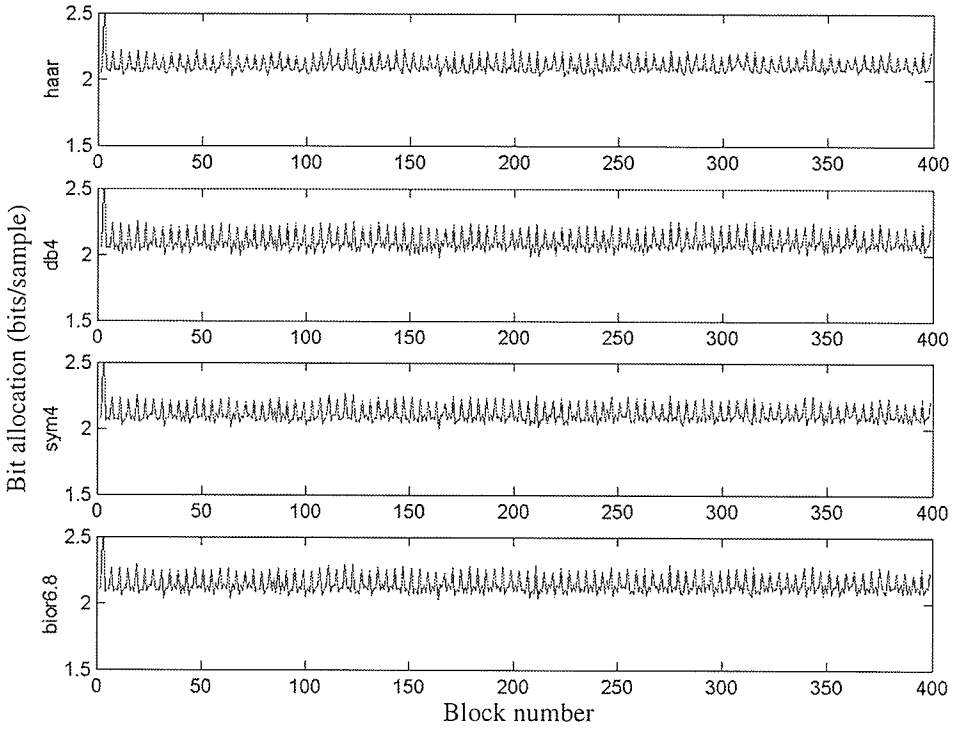


Fig. E.29. A_1 bit allocation from the R1-24919PS test set with Haar, db4, sym4, and bior6.8 wavelets.

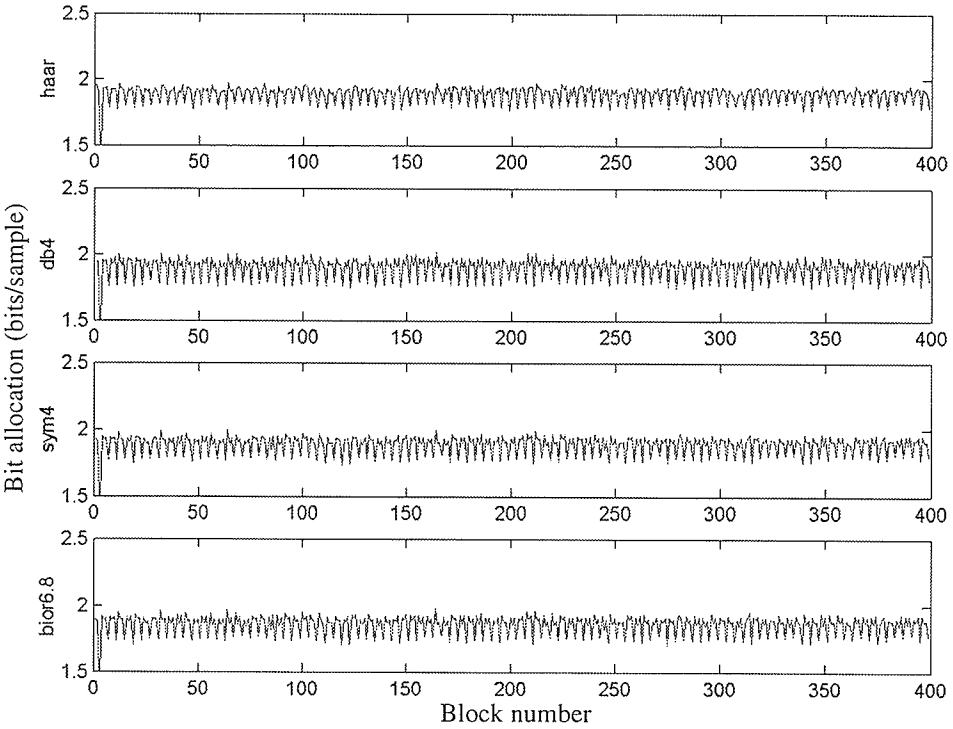


Fig. E.30. D_1 bit allocation from the R1-24919PS test set with Haar, db4, sym4, and bior6.8 wavelets.

E.3 Variance Ratio Results

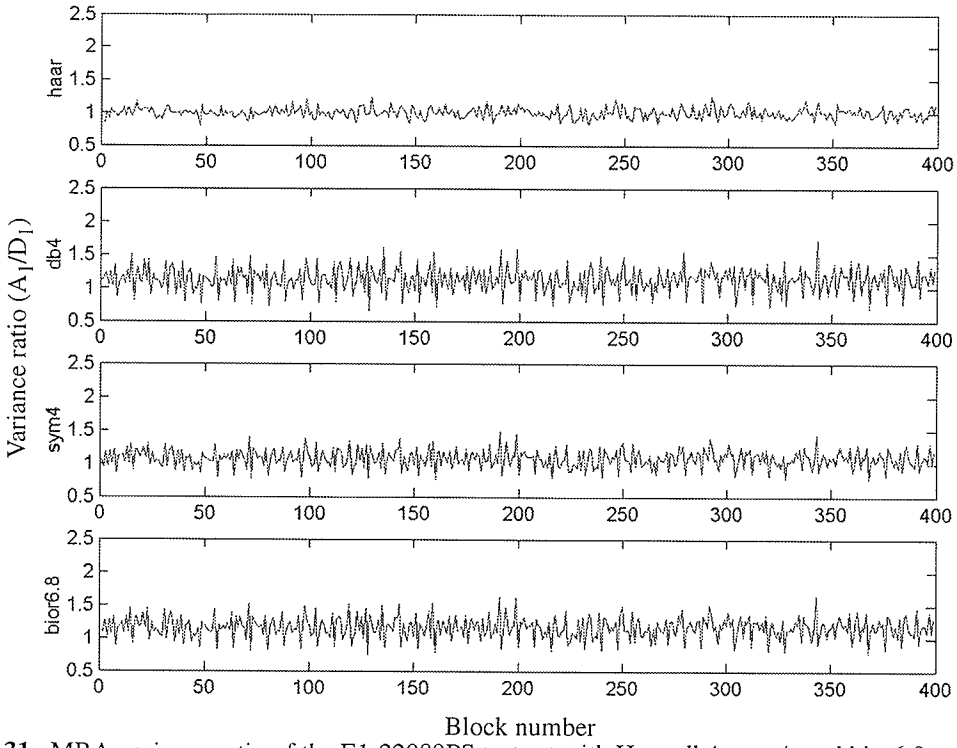


Fig. E.31. MRA variance ratio of the E1-22089PS test set with Haar, db4, sym4, and bior6.8 wavelets.

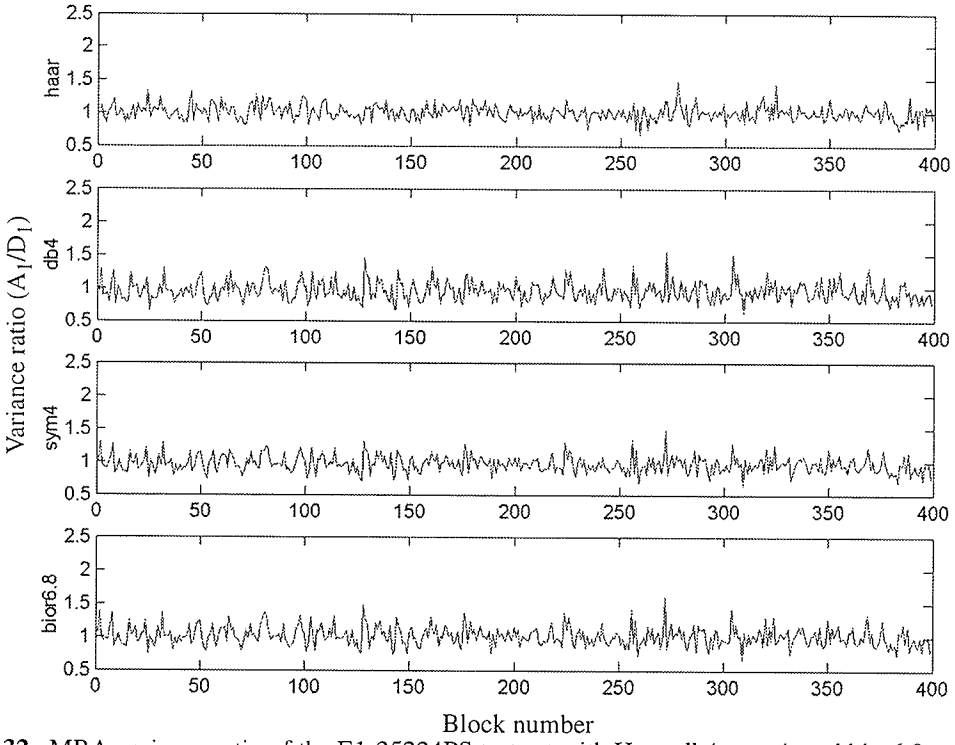


Fig. E.32. MRA variance ratio of the E1-25224PS test set with Haar, db4, sym4, and bior6.8 wavelets.

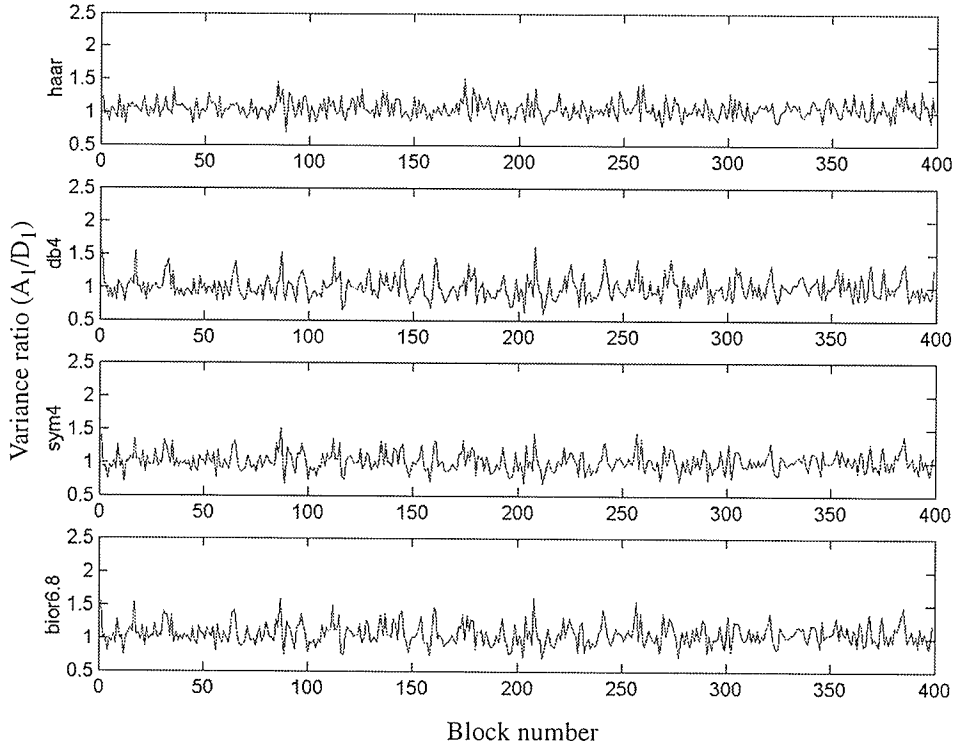


Fig. E.33. MRA variance ratio of the E2-5551PS test set with Haar, db4, sym4, and bior6.8 wavelets.

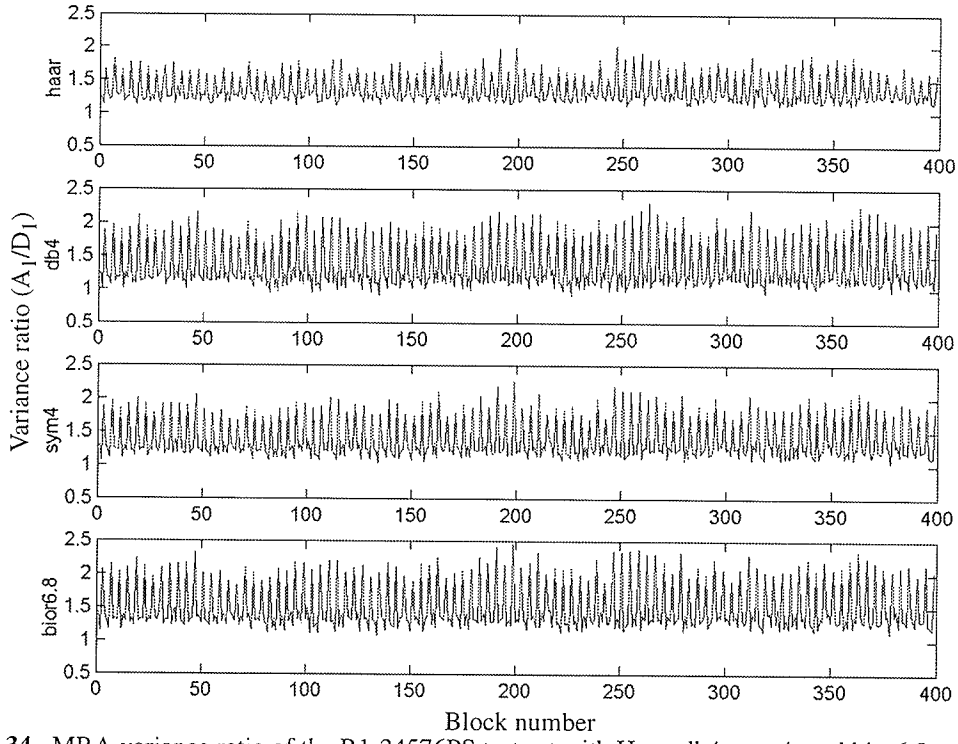


Fig. E.34. MRA variance ratio of the R1-24576PS test set with Haar, db4, sym4, and bior6.8 wavelets.

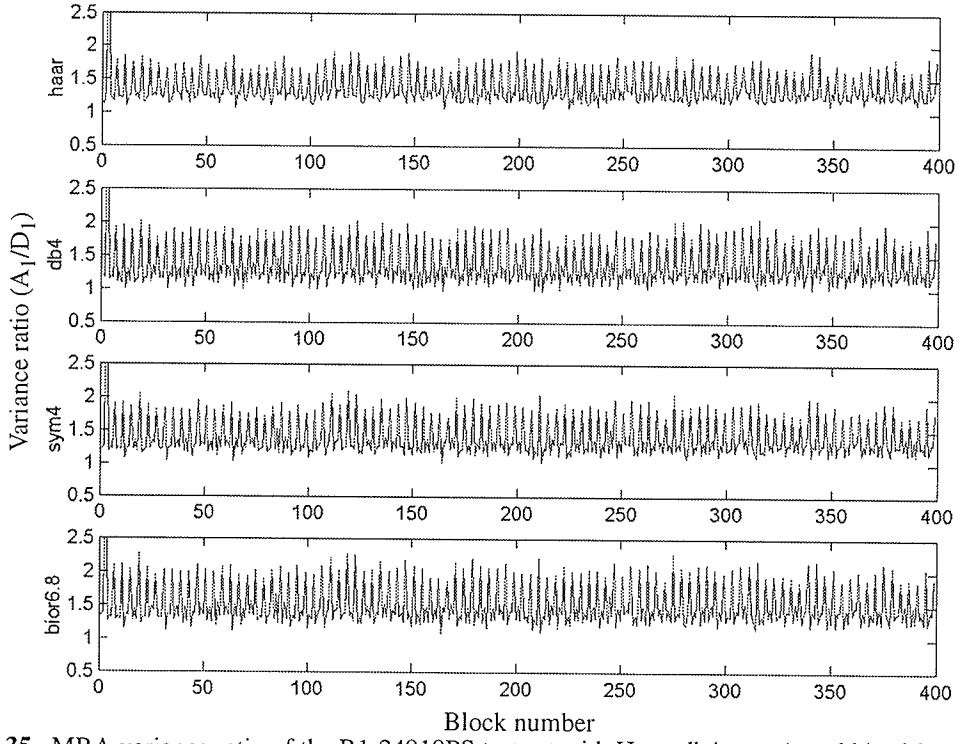


Fig. E.35. MRA variance ratio of the R1-24919PS test set with Haar, db4, sym4, and bior6.8 wavelets.

E.4 SQNR Results for Standard Wavlet Basis

Table E.1 SQNR of all standard wavelet basis in Matlab for the first 200 blocks of the E1-25224PS, E2-5551PS, and R1-24919PS compared to the Shannon Bound, BAQ, and the wavelet mean.

Wavelet basis	Test set SQNR in dB								
	E1-25224PS			E2-5551PS			R1-24919PS		
	2	3	4	2	3	4	2	3	4
Shannon bound	12.04	18.06	24.08	12.04	18.06	24.08	12.04	18.06	24.08
BAQ	9.80	15.32	20.58	9.36	14.66	20.19	9.45	15.02	20.42
Wavelet mean	9.13	14.32	19.75	9.01	14.15	19.56	8.70	13.83	19.48
haar	9.59	14.96	20.56	9.42	14.74	20.30	9.12	14.44	20.32
demy	9.53	14.90	20.49	9.40	14.72	20.28	9.08	14.39	20.25
db2	9.61	14.99	20.55	9.42	14.71	20.30	9.15	14.48	20.36
db3	9.57	14.93	20.52	9.43	14.70	20.28	9.11	14.42	20.33
db4	9.54	14.89	20.50	9.42	14.70	20.27	9.10	14.41	20.31
db5	9.55	14.91	20.51	9.40	14.72	20.28	9.10	14.41	20.29
db6	9.53	14.90	20.50	9.40	14.73	20.30	9.08	14.39	20.27
db7	9.53	14.88	20.48	9.40	14.71	20.29	9.07	14.38	20.23
db8	9.52	14.88	20.48	9.40	14.70	20.29	9.07	14.37	20.23
db9	9.51	14.89	20.47	9.40	14.70	20.26	9.07	14.35	20.22
db10	9.51	14.87	20.47	9.40	14.70	20.25	9.06	14.35	20.22
db11	9.50	14.87	20.43	9.40	14.70	20.27	9.06	14.35	20.23
db12	9.51	14.86	20.44	9.40	14.70	20.26	9.06	14.35	20.22
db13	9.51	14.83	20.44	9.40	14.71	20.26	9.06	14.36	20.21
db14	9.50	14.85	20.44	9.40	14.71	20.28	9.06	14.35	20.20
db15	9.50	14.85	20.45	9.39	14.70	20.30	9.06	14.33	20.19
db16	9.50	14.86	20.46	9.40	14.70	20.31	9.05	14.31	20.16

Wavelet basis	Test set SQNR in dB								
	E1-25224PS			E2-5551PS			R1-24919PS		
	2	3	4	2	3	4	2	3	4
Shannon bound	12.04	18.06	24.08	12.04	18.06	24.08	12.04	18.06	24.08
BAQ	9.80	15.32	20.58	9.36	14.66	20.19	9.45	15.02	20.42
Wavelet mean	9.13	14.32	19.75	9.01	14.15	19.56	8.70	13.83	19.48
db17	9.50	14.87	20.49	9.40	14.70	20.31	9.04	14.32	20.16
db18	9.50	14.88	20.51	9.39	14.68	20.29	9.04	14.31	20.16
db19	9.49	14.85	20.48	9.39	14.68	20.27	9.04	14.32	20.17
db20	9.50	14.86	20.44	9.39	14.69	20.26	9.04	14.32	20.18
db21	9.51	14.86	20.42	9.40	14.70	20.28	9.05	14.33	20.17
db22	9.50	14.83	20.42	9.41	14.70	20.27	9.05	14.33	20.17
db23	9.50	14.85	20.44	9.41	14.71	20.27	9.05	14.33	20.17
db24	9.49	14.84	20.44	9.40	14.70	20.30	9.05	14.33	20.16
db25	9.49	14.83	20.47	9.40	14.70	20.31	9.04	14.32	20.16
db26	9.50	14.85	20.46	9.40	14.71	20.31	9.04	14.30	20.14
db27	9.49	14.84	20.48	9.40	14.70	20.29	9.03	14.31	20.14
db28	9.49	14.85	20.47	9.39	14.68	20.28	9.04	14.30	20.13
db29	9.49	14.84	20.46	9.39	14.68	20.25	9.04	14.31	20.15
db30	9.49	14.84	20.43	9.41	14.71	20.28	9.04	14.31	20.15
db31	9.48	14.84	20.40	9.41	14.72	20.29	9.04	14.32	20.15
db32	9.49	14.84	20.41	9.41	14.70	20.27	9.05	14.31	20.15
db33	9.49	14.84	20.44	9.41	14.70	20.28	9.04	14.32	20.15
db34	9.48	14.83	20.42	9.40	14.69	20.30	9.03	14.31	20.14
db35	9.48	14.82	20.45	9.39	14.69	20.32	9.03	14.30	20.13
db36	9.48	14.81	20.47	9.40	14.70	20.30	9.02	14.29	20.12

Wavelet basis	Test set SQNR in dB								
	E1-25224PS			E2-5551PS			R1-24919PS		
	2	3	4	2	3	4	2	3	4
Shannon bound	12.04	18.06	24.08	12.04	18.06	24.08	12.04	18.06	24.08
BAQ	9.80	15.32	20.58	9.36	14.66	20.19	9.45	15.02	20.42
Wavelet mean	9.13	14.32	19.75	9.01	14.15	19.56	8.70	13.83	19.48
db37	9.47	14.83	20.45	9.40	14.70	20.29	9.03	14.29	20.11
db38	9.47	14.82	20.43	9.40	14.69	20.27	9.03	14.29	20.12
db39	9.46	14.82	20.44	9.40	14.69	20.26	9.03	14.29	20.12
db40	9.46	14.81	20.42	9.40	14.71	20.27	9.03	14.30	20.12
db41	9.47	14.82	20.43	9.40	14.71	20.29	9.04	14.30	20.13
db42	9.48	14.83	20.43	9.41	14.70	20.26	9.03	14.29	20.12
db43	9.47	14.83	20.43	9.40	14.69	20.29	9.02	14.29	20.11
db44	9.46	14.81	20.40	9.40	14.68	20.30	9.02	14.28	20.10
sym2	9.61	14.99	20.55	9.42	14.71	20.30	9.15	14.48	20.36
sym3	9.57	14.93	20.52	9.43	14.70	20.28	9.11	14.42	20.33
sym4	9.59	14.94	20.50	9.42	14.73	20.31	9.10	14.42	20.31
sym5	9.56	14.92	20.50	9.40	14.71	20.29	9.10	14.41	20.30
sym6	9.57	14.95	20.52	9.41	14.73	20.29	9.09	14.41	20.30
sym7	9.56	14.92	20.53	9.41	14.71	20.29	9.09	14.40	20.29
sym8	9.57	14.94	20.52	9.41	14.73	20.28	9.09	14.40	20.29
sym9	9.55	14.90	20.50	9.40	14.71	20.28	9.09	14.40	20.28
sym10	9.56	14.94	20.51	9.41	14.73	20.29	9.09	14.39	20.28
sym11	9.55	14.91	20.51	9.40	14.71	20.29	9.08	14.38	20.26
sym12	9.54	14.88	20.51	9.40	14.71	20.27	9.09	14.39	20.28
sym13	9.55	14.89	20.48	9.41	14.72	20.29	9.09	14.40	20.29

Wavelet basis	Test set SQNR in dB								
	E1-25224PS			E2-5551PS			R1-24919PS		
	2	3	4	2	3	4	2	3	4
Shannon bound	12.04	18.06	24.08	12.04	18.06	24.08	12.04	18.06	24.08
BAQ	9.80	15.32	20.58	9.36	14.66	20.19	9.45	15.02	20.42
Wavelet mean	9.13	14.32	19.75	9.01	14.15	19.56	8.70	13.83	19.48
sym14	9.54	14.88	20.49	9.41	14.71	20.28	9.09	14.39	20.28
sym15	9.54	14.90	20.49	9.40	14.70	20.29	9.08	14.38	20.26
sym16	9.54	14.88	20.50	9.41	14.71	20.28	9.09	14.39	20.27
sym17	9.54	14.89	20.49	9.41	14.71	20.26	9.08	14.38	20.27
sym18	9.54	14.91	20.49	9.41	14.72	20.28	9.08	14.39	20.27
sym19	9.53	14.89	20.49	9.40	14.70	20.30	9.08	14.38	20.26
sym20	9.54	14.88	20.50	9.41	14.72	20.29	9.09	14.39	20.27
sym21	9.54	14.89	20.50	9.41	14.71	20.27	9.08	14.38	20.27
sym22	9.54	14.88	20.50	9.41	14.72	20.29	9.09	14.39	20.27
sym23	9.53	14.89	20.50	9.40	14.70	20.31	9.08	14.38	20.25
coif1	9.61	14.97	20.56	9.42	14.74	20.31	9.12	14.46	20.35
coif2	9.59	14.96	20.53	9.41	14.73	20.31	9.10	14.43	20.32
coif3	9.57	14.95	20.52	9.42	14.73	20.30	9.09	14.41	20.30
coif4	9.56	14.94	20.52	9.42	14.72	20.29	9.09	14.41	20.29
coif5	9.55	14.93	20.51	9.41	14.73	20.28	9.09	14.40	20.28
bior1.3	9.50	14.83	20.43	9.32	14.62	20.19	9.01	14.34	20.23
bior1.5	9.39	14.72	20.32	9.22	14.51	20.07	8.92	14.25	20.13
bior2.2	9.33	14.68	20.26	9.17	14.46	20.02	9.05	14.51	20.29
bior2.4	9.40	14.77	20.34	9.25	14.55	20.11	9.10	14.54	20.33
bior2.6	9.39	14.74	20.31	9.23	14.53	20.08	9.08	14.52	20.30

Wavelet basis	Test set SQNR in dB								
	E1-25224PS			E2-5551PS			R1-24919PS		
	2	3	4	2	3	4	2	3	4
Shannon bound	12.04	18.06	24.08	12.04	18.06	24.08	12.04	18.06	24.08
BAQ	9.80	15.32	20.58	9.36	14.66	20.19	9.45	15.02	20.42
Wavelet mean	9.13	14.32	19.75	9.01	14.15	19.56	8.70	13.83	19.48
bior2.8	9.35	14.71	20.30	9.20	14.50	20.05	9.05	14.49	20.26
bior3.1	7.84	13.07	18.61	7.77	12.93	18.45	7.99	13.33	18.88
bior3.3	8.62	13.89	19.48	8.53	13.74	19.28	8.56	13.97	19.41
bior3.5	8.79	14.09	19.67	8.69	13.93	19.46	8.69	14.11	19.53
bior3.7	8.83	14.13	19.72	8.73	13.97	19.50	8.72	14.15	19.57
bior3.9	8.85	14.15	19.72	8.73	13.98	19.52	8.73	14.15	19.56
bior4.4	9.50	14.86	20.42	9.34	14.65	20.23	9.06	14.38	20.27
bior5.5	9.38	14.71	20.31	9.22	14.50	20.08	8.85	14.03	19.74
bior6.8	9.53	14.92	20.49	9.39	14.70	20.26	9.10	14.44	20.33
rbio1.3	9.42	14.77	20.36	9.26	14.56	20.13	9.01	14.32	20.20
rbio1.5	9.29	14.62	20.22	9.13	14.42	20.00	8.90	14.20	20.06
rbio2.2	9.34	14.66	20.24	9.14	14.43	19.99	8.71	13.90	19.60
rbio2.4	9.38	14.69	20.28	9.19	14.48	20.05	8.75	13.94	19.63
rbio2.6	9.35	14.66	20.25	9.16	14.45	20.04	8.72	13.91	19.60
rbio2.8	9.31	14.61	20.20	9.13	14.42	20.00	8.69	13.87	19.56
rbio3.1	7.81	13.02	18.54	7.69	12.82	18.36	7.14	12.19	17.71
rbio3.3	8.59	13.84	19.41	8.43	13.61	19.19	7.84	12.92	18.45
rbio3.5	8.76	14.04	19.58	8.59	13.80	19.35	8.02	13.10	18.63
rbio3.7	8.82	14.09	19.64	8.63	13.85	19.41	8.07	13.14	18.67
rbio3.9	8.82	14.09	19.66	8.64	13.86	19.42	8.08	13.15	18.69

Wavelet basis	Test set SQNR in dB								
	E1-25224PS			E2-5551PS			R1-24919PS		
	2	3	4	2	3	4	2	3	4
Shannon bound	12.04	18.06	24.08	12.04	18.06	24.08	12.04	18.06	24.08
BAQ	9.80	15.32	20.58	9.36	14.66	20.19	9.45	15.02	20.42
Wavelet mean	9.13	14.32	19.75	9.01	14.15	19.56	8.70	13.83	19.48
rbio4.4	9.53	14.88	20.46	9.35	14.65	20.22	9.01	14.34	20.22
rbio5.5	9.43	14.77	20.36	9.29	14.55	20.14	9.08	14.51	20.31
rbio6.8	9.55	14.92	20.49	9.39	14.69	20.26	9.03	14.33	20.20

Table E.2 SQNR of all wavelets in Matlab for the first 200 blocks of the E1-22089PS, R1-24576PS, and the mean of all test sets compared to the Shannon Bound BAQ, and the wavelet mean.

Wavelet basis	Test set SQNR in dB								
	E1-22089PS			R1-24576PS			Mean of all test sets		
	2	3	4	2	3	4	2	3	4
Shannon bound	12.04	18.06	24.08	12.04	18.06	24.08	12.04	18.06	24.08
BAQ	9.60	15.00	20.51	9.45	15.02	20.42	9.52	15.00	20.42
Wavelet mean	9.24	14.56	20.16	8.94	14.28	20.09	9.17	14.49	20.17
haar	9.40	14.74	20.36	9.12	14.49	20.40	9.33	14.67	20.39
demy	9.35	14.70	20.30	9.05	14.41	20.28	9.28	14.62	20.32
db2	9.40	14.75	20.36	9.13	14.53	20.43	9.34	14.69	20.40
db3	9.38	14.71	20.32	9.08	14.47	20.38	9.32	14.65	20.37
db4	9.37	14.69	20.30	9.07	14.45	20.33	9.30	14.63	20.34
db5	9.36	14.70	20.31	9.08	14.45	20.32	9.30	14.64	20.34

Wavelet basis	Test set SQNR in dB								
	E1-22089PS			R1-24576PS			Mean of all test sets		
	2	3	4	2	3	4	2	3	4
Shannon bound	12.04	18.06	24.08	12.04	18.06	24.08	12.04	18.06	24.08
BAQ	9.60	15.00	20.51	9.45	15.02	20.42	9.52	15.00	20.42
Wavelet mean	9.24	14.56	20.16	8.94	14.28	20.09	9.17	14.49	20.17
db6	9.36	14.70	20.30	9.07	14.43	20.28	9.29	14.63	20.33
db7	9.35	14.70	20.28	9.06	14.40	20.28	9.28	14.61	20.31
db8	9.35	14.69	20.26	9.05	14.40	20.28	9.28	14.61	20.31
db9	9.36	14.69	20.27	9.04	14.40	20.27	9.28	14.61	20.30
db10	9.37	14.70	20.29	9.02	14.38	20.24	9.27	14.60	20.29
db11	9.36	14.69	20.30	9.01	14.37	20.23	9.27	14.60	20.29
db12	9.36	14.69	20.30	9.02	14.37	20.23	9.27	14.59	20.29
db13	9.35	14.67	20.30	9.03	14.36	20.21	9.27	14.59	20.28
db14	9.35	14.66	20.27	9.03	14.36	20.20	9.27	14.59	20.28
db15	9.34	14.67	20.27	9.03	14.35	20.19	9.26	14.58	20.28
db16	9.33	14.67	20.26	9.02	14.35	20.18	9.26	14.58	20.27
db17	9.34	14.67	20.25	9.02	14.35	20.20	9.26	14.58	20.28
db18	9.34	14.69	20.27	9.01	14.34	20.20	9.26	14.58	20.28
db19	9.36	14.71	20.28	9.01	14.35	20.20	9.26	14.58	20.28
db20	9.37	14.70	20.31	9.01	14.35	20.19	9.26	14.58	20.27
db21	9.36	14.68	20.29	9.01	14.35	20.18	9.26	14.58	20.27
db22	9.35	14.66	20.30	9.01	14.34	20.16	9.26	14.57	20.26
db23	9.34	14.65	20.28	9.01	14.33	20.16	9.26	14.57	20.26
db24	9.34	14.66	20.26	9.01	14.33	20.14	9.26	14.57	20.26
db25	9.33	14.67	20.27	9.01	14.34	20.16	9.26	14.57	20.27

Wavelet basis	Test set SQNR in dB								
	E1-22089PS			R1-24576PS			Mean of all test sets		
	2	3	4	2	3	4	2	3	4
Shannon bound	12.04	18.06	24.08	12.04	18.06	24.08	12.04	18.06	24.08
BAQ	9.60	15.00	20.51	9.45	15.02	20.42	9.52	15.00	20.42
Wavelet mean	9.24	14.56	20.16	8.94	14.28	20.09	9.17	14.49	20.17
db26	9.34	14.68	20.25	9.02	14.34	20.17	9.26	14.58	20.27
db27	9.34	14.67	20.26	9.01	14.34	20.17	9.25	14.57	20.27
db28	9.34	14.69	20.26	9.01	14.34	20.18	9.25	14.57	20.27
db29	9.35	14.68	20.27	9.01	14.34	20.17	9.26	14.57	20.26
db30	9.35	14.68	20.26	9.01	14.33	20.16	9.26	14.58	20.26
db31	9.34	14.66	20.27	9.00	14.32	20.14	9.26	14.57	20.25
db32	9.34	14.66	20.28	9.00	14.32	20.13	9.26	14.57	20.25
db33	9.33	14.64	20.27	9.01	14.32	20.12	9.25	14.56	20.25
db34	9.33	14.65	20.27	9.01	14.32	20.12	9.25	14.56	20.25
db35	9.33	14.66	20.25	9.01	14.32	20.14	9.25	14.56	20.26
db36	9.33	14.66	20.24	9.01	14.33	20.15	9.25	14.56	20.26
db37	9.33	14.67	20.25	9.01	14.33	20.16	9.25	14.56	20.25
db38	9.33	14.67	20.25	9.00	14.32	20.13	9.25	14.56	20.24
db39	9.34	14.65	20.25	9.00	14.31	20.12	9.25	14.55	20.24
db40	9.34	14.65	20.24	8.99	14.30	20.11	9.25	14.55	20.23
db41	9.34	14.64	20.26	8.99	14.30	20.10	9.25	14.56	20.24
db42	9.33	14.64	20.25	8.99	14.30	20.09	9.25	14.55	20.23
db43	9.32	14.64	20.26	8.99	14.30	20.10	9.24	14.55	20.24
db44	9.32	14.65	20.26	8.99	14.31	20.10	9.24	14.55	20.23
sym2	9.40	14.75	20.36	9.13	14.53	20.43	9.34	14.69	20.40

Wavelet basis	Test set SQNR in dB								
	E1-22089PS			R1-24576PS			Mean of all test sets		
	2	3	4	2	3	4	2	3	4
Shannon bound	12.04	18.06	24.08	12.04	18.06	24.08	12.04	18.06	24.08
BAQ	9.60	15.00	20.51	9.45	15.02	20.42	9.52	15.00	20.42
Wavelet mean	9.24	14.56	20.16	8.94	14.28	20.09	9.17	14.49	20.17
sym3	9.38	14.71	20.32	9.08	14.47	20.38	9.32	14.65	20.37
sym4	9.37	14.72	20.33	9.09	14.46	20.37	9.31	14.65	20.37
sym5	9.38	14.72	20.31	9.07	14.44	20.31	9.30	14.64	20.34
sym6	9.36	14.71	20.33	9.08	14.46	20.35	9.30	14.65	20.36
sym7	9.38	14.69	20.29	9.06	14.43	20.31	9.30	14.63	20.34
sym8	9.36	14.71	20.32	9.07	14.45	20.33	9.30	14.65	20.35
sym9	9.37	14.70	20.32	9.06	14.43	20.30	9.30	14.63	20.33
sym10	9.36	14.70	20.32	9.07	14.44	20.32	9.30	14.64	20.34
sym11	9.36	14.69	20.30	9.05	14.40	20.28	9.29	14.62	20.33
sym12	9.37	14.69	20.30	9.06	14.42	20.30	9.29	14.62	20.33
sym13	9.37	14.70	20.31	9.06	14.42	20.30	9.30	14.63	20.34
sym14	9.37	14.69	20.31	9.06	14.42	20.30	9.29	14.62	20.33
sym15	9.36	14.69	20.31	9.05	14.40	20.27	9.29	14.62	20.32
sym16	9.37	14.70	20.31	9.06	14.42	20.29	9.29	14.62	20.33
sym17	9.36	14.69	20.32	9.05	14.41	20.29	9.29	14.62	20.33
sym18	9.36	14.71	20.29	9.06	14.43	20.30	9.29	14.63	20.33
sym19	9.36	14.70	20.31	9.05	14.40	20.27	9.28	14.61	20.33
sym20	9.37	14.70	20.30	9.06	14.42	20.29	9.29	14.62	20.33
sym21	9.36	14.69	20.32	9.05	14.41	20.29	9.29	14.62	20.33
sym22	9.36	14.70	20.31	9.06	14.41	20.29	9.29	14.62	20.33

Wavelet basis	Test set SQNR in dB								
	E1-22089PS			R1-24576PS			Mean of all test sets		
	2	3	4	2	3	4	2	3	4
Shannon bound	12.04	18.06	24.08	12.04	18.06	24.08	12.04	18.06	24.08
BAQ	9.60	15.00	20.51	9.45	15.02	20.42	9.52	15.00	20.42
Wavelet mean	9.24	14.56	20.16	8.94	14.28	20.09	9.17	14.49	20.17
sym23	9.36	14.69	20.31	9.05	14.40	20.27	9.28	14.61	20.33
coif1	9.39	14.75	20.36	9.13	14.51	20.41	9.33	14.69	20.40
coif2	9.37	14.73	20.34	9.10	14.48	20.36	9.31	14.66	20.37
coif3	9.37	14.71	20.33	9.08	14.46	20.33	9.31	14.65	20.36
coif4	9.36	14.71	20.32	9.07	14.45	20.32	9.30	14.65	20.35
coif5	9.36	14.71	20.32	9.07	14.44	20.31	9.30	14.64	20.34
bior1.3	9.22	14.55	20.17	9.01	14.39	20.28	9.21	14.55	20.26
bior1.5	9.08	14.40	20.02	8.92	14.30	20.17	9.11	14.44	20.14
bior2.2	9.15	14.49	20.09	9.06	14.53	20.14	9.15	14.54	20.16
bior2.4	9.19	14.53	20.13	9.10	14.58	20.17	9.21	14.59	20.22
bior2.6	9.16	14.50	20.09	9.08	14.55	20.12	9.19	14.57	20.18
bior2.8	9.12	14.46	20.05	9.05	14.52	20.07	9.16	14.54	20.15
bior3.1	7.66	12.82	18.37	7.96	13.22	18.80	7.84	13.07	18.62
bior3.3	8.42	13.64	19.21	8.54	13.91	19.45	8.53	13.83	19.36
bior3.5	8.59	13.83	19.41	8.67	14.06	19.58	8.68	14.01	19.53
bior3.7	8.64	13.89	19.45	8.70	14.10	19.61	8.72	14.05	19.57
bior3.9	8.65	13.91	19.47	8.70	14.10	19.61	8.73	14.06	19.58
bior4.4	9.32	14.67	20.28	9.05	14.42	20.30	9.25	14.60	20.30
bior5.5	9.22	14.53	20.14	8.81	14.05	19.85	9.10	14.36	20.02
bior6.8	9.35	14.70	20.30	9.09	14.48	20.34	9.29	14.65	20.34

Wavelet basis	Test set SQNR in dB								
	E1-22089PS			R1-24576PS			Mean of all test sets		
	2	3	4	2	3	4	2	3	4
Shannon bound	12.04	18.06	24.08	12.04	18.06	24.08	12.04	18.06	24.08
BAQ	9.60	15.00	20.51	9.45	15.02	20.42	9.52	15.00	20.42
Wavelet mean	9.24	14.56	20.16	8.94	14.28	20.09	9.17	14.49	20.17
rbio1.3	9.32	14.65	20.27	9.00	14.35	20.24	9.20	14.53	20.24
rbio1.5	9.22	14.54	20.14	8.88	14.21	20.10	9.08	14.40	20.10
rbio2.2	9.09	14.43	20.04	8.71	13.96	19.74	9.00	14.28	19.92
rbio2.4	9.17	14.50	20.11	8.75	13.98	19.76	9.05	14.32	19.97
rbio2.6	9.16	14.48	20.09	8.72	13.94	19.72	9.02	14.29	19.94
rbio2.8	9.12	14.43	20.06	8.68	13.90	19.67	8.99	14.25	19.90
rbio3.1	7.65	12.83	18.42	7.12	12.19	17.79	7.48	12.61	18.16
rbio3.3	8.40	13.62	19.24	7.84	12.93	18.56	8.22	13.38	18.97
rbio3.5	8.56	13.79	19.42	8.00	13.10	18.73	8.39	13.56	19.14
rbio3.7	8.61	13.84	19.47	8.05	13.15	18.77	8.44	13.61	19.19
rbio3.9	8.62	13.85	19.48	8.06	13.16	18.78	8.45	13.62	19.21
rbio4.4	9.28	14.63	20.23	9.01	14.39	20.26	9.23	14.58	20.28
rbio5.5	9.20	14.51	20.10	9.06	14.52	20.11	9.21	14.57	20.20
rbio6.8	9.33	14.68	20.29	9.02	14.37	20.25	9.26	14.60	20.30

APPENDIX F
RESULTS OF FPGA BAQ AND WAVELET-BAQ

F.1 E1-22089PS Test Set Histograms

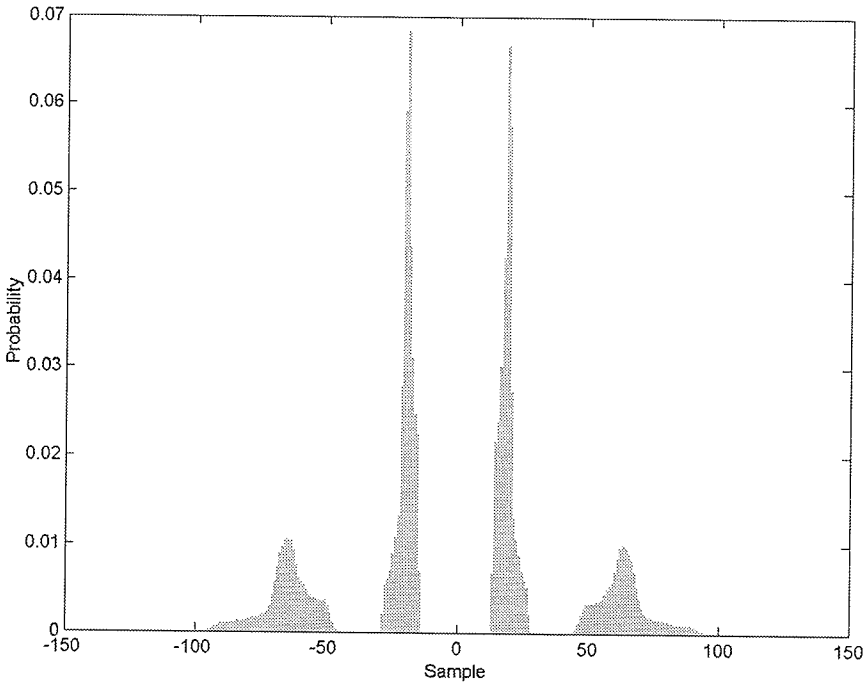


Fig. F.1. Histogram of the E1-22089PS test set quantized with MatLab BAQ-2.

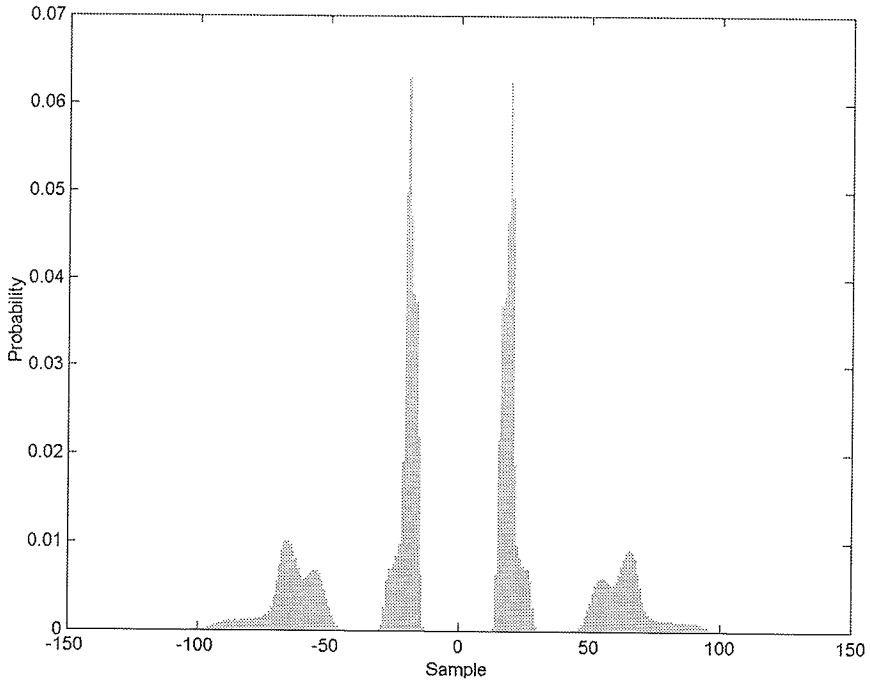


Fig. F.2. Histogram of the E1-22089PS test set quantized with FPGA BAQ-2.

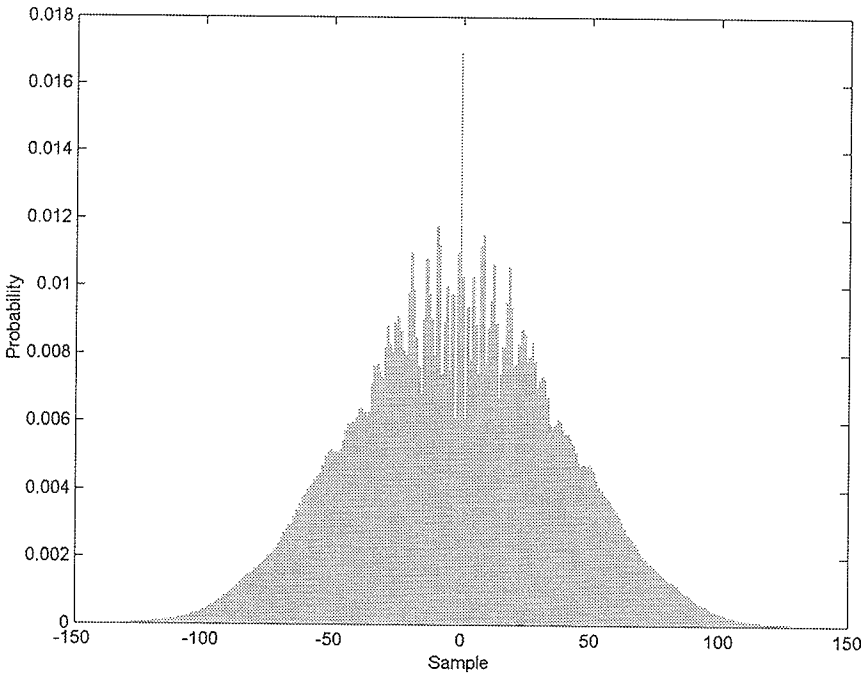


Fig. F.3. Histogram of the E1-22089PS test set quantized with FPGA Wavelet-BAQ-2.

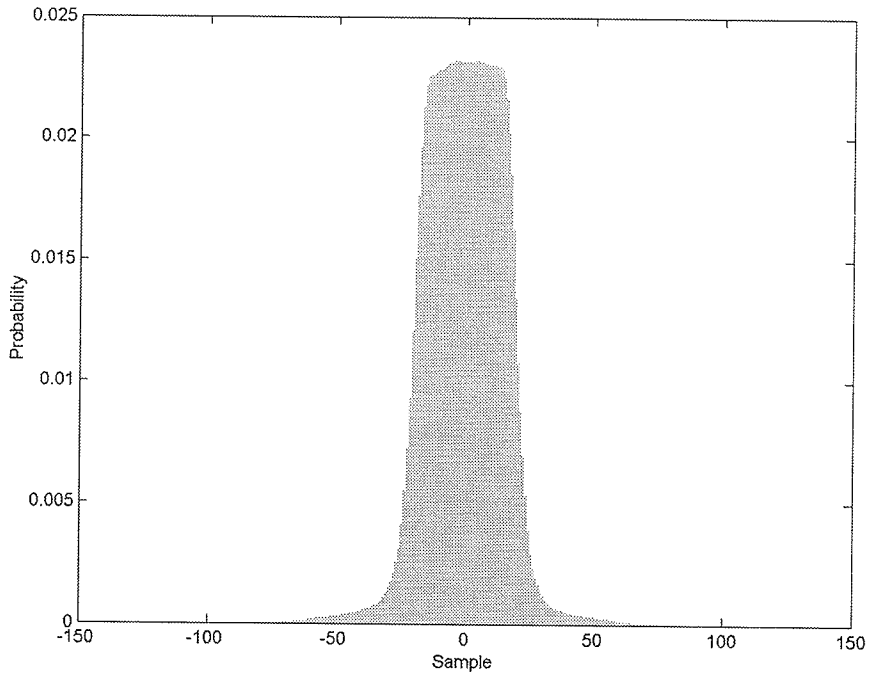


Fig. F.4. Error histogram of the E1-22089PS test set quantized with FPGA BAQ-2.

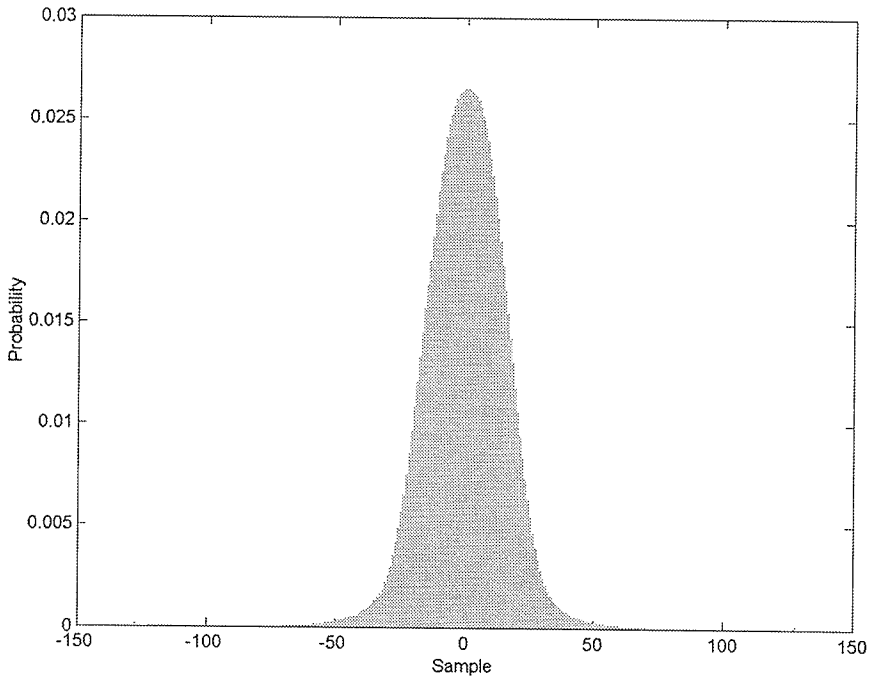


Fig. F.5. Error Histogram of the E1-22089PS test set quantized with FPGA Wavelet-BAQ-2.

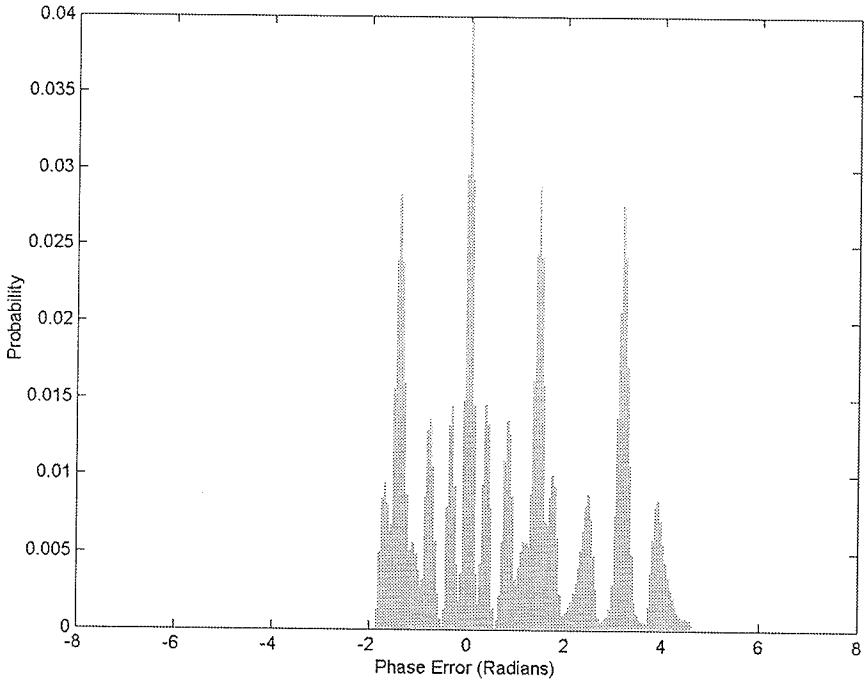


Fig. F.6. Phase error histogram of the E1-22089PS test set quantized with FPGA BAQ-2.

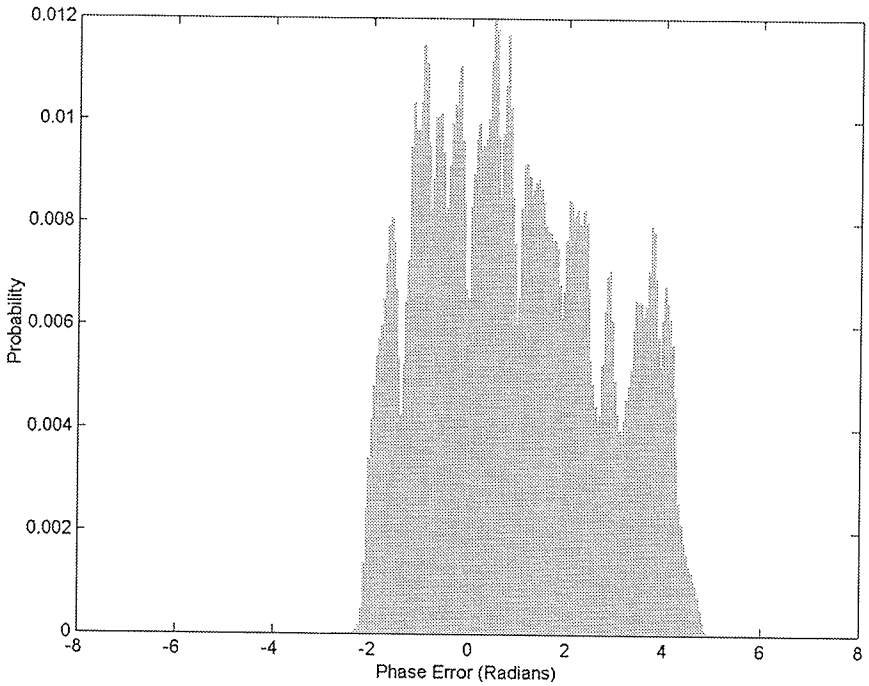


Fig. F.7. Phase error histogram of the E1-22089PS test set quantized with FPGA Wavelet-BAQ-2.

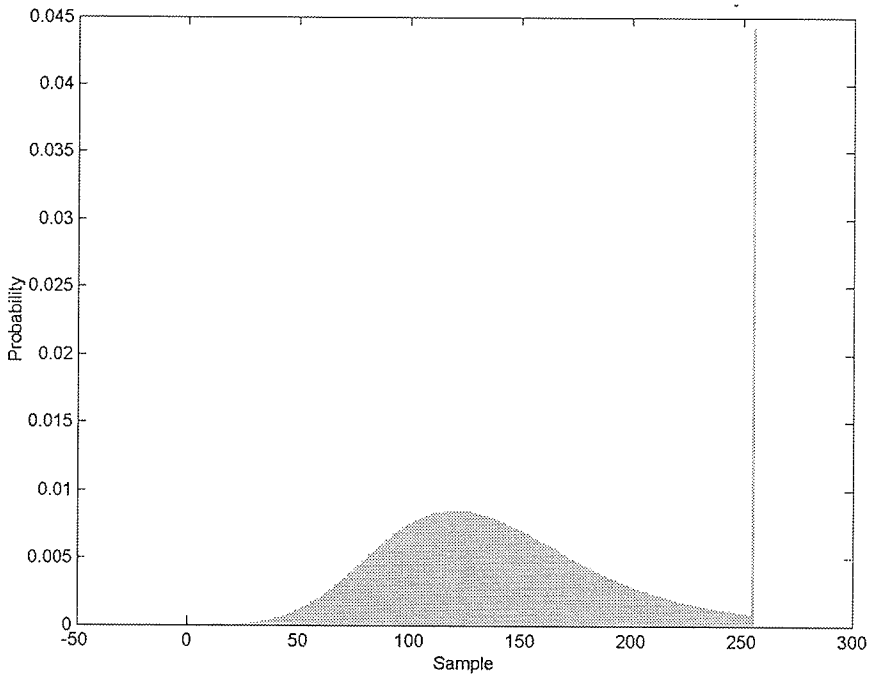


Fig. F.8. Image histogram of the E1-22089PS test set quantized with FPGA BAQ-2.

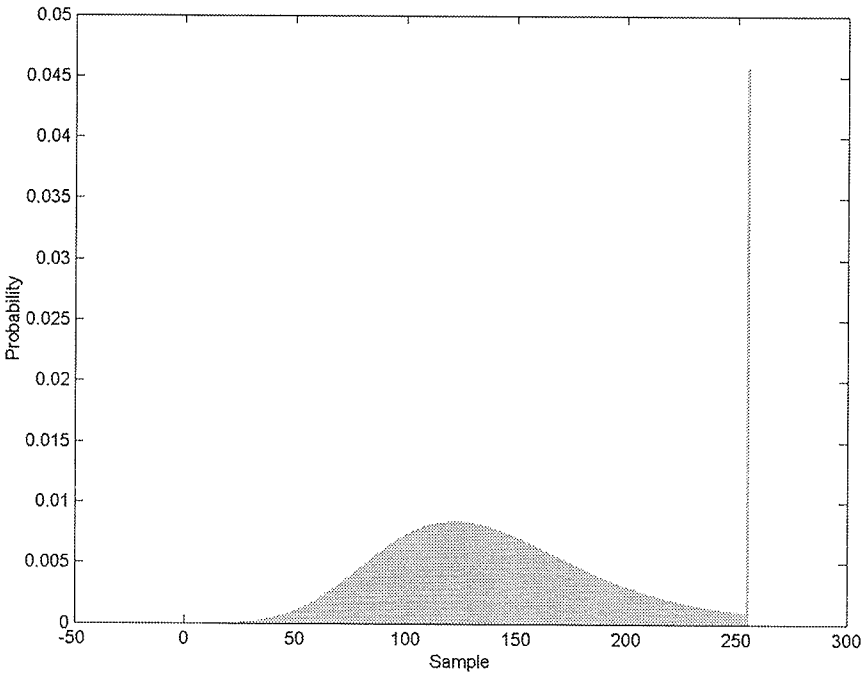


Fig. F.9. Image histogram of the E1-22089PS test set quantized with FPGA Wavelet-BAQ-2.

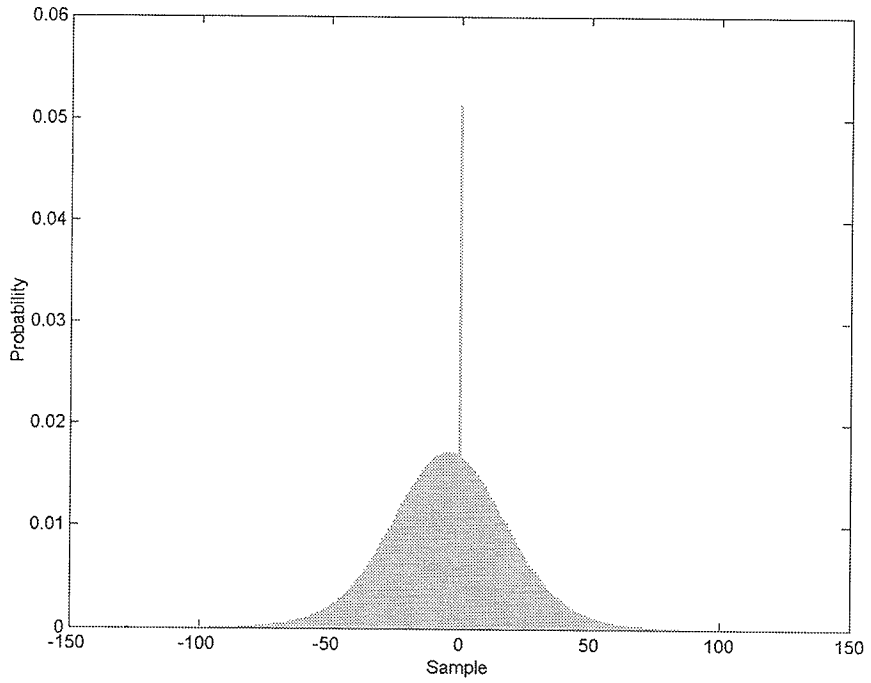


Fig. F.10. Image error histogram of the E1-22089PS test set quantized with FPGA BAQ-2.

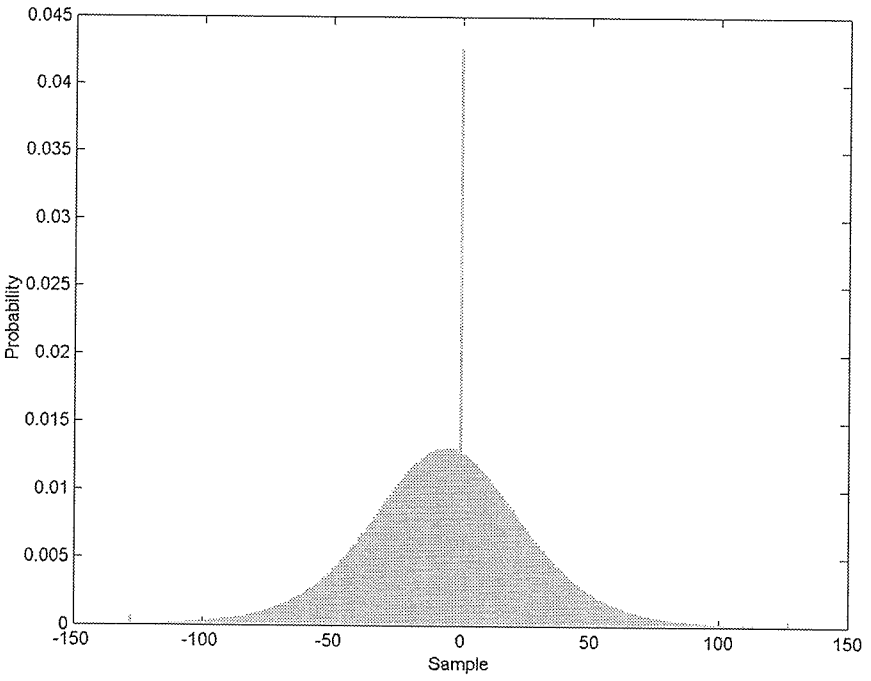


Fig. F.11. Image error histogram of the E1-22089PS test set quantized with FPGA Wavelet-BAQ-2.

F.2 E1-25224PS Test Set Histograms

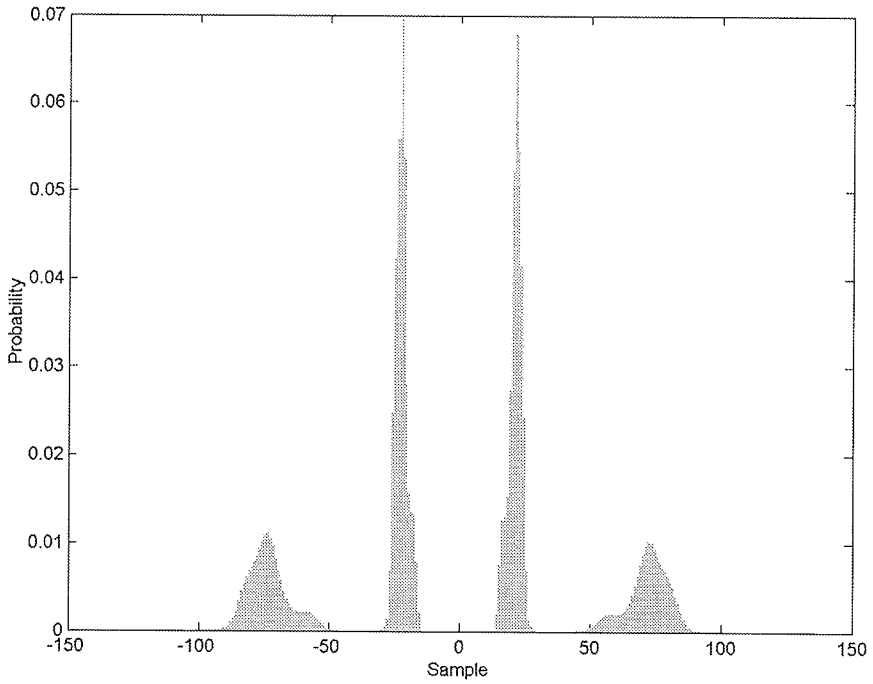


Fig. F.12. Histogram of the E1-25224PS test set quantized with MatLab BAQ-2.

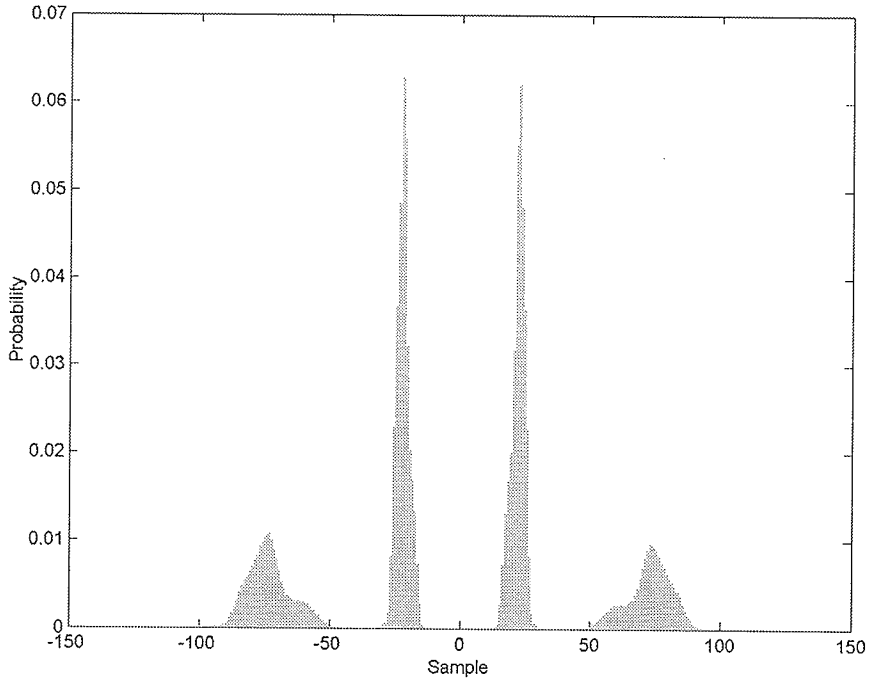


Fig. F.13. Histogram of the E1-25224PS test set quantized with FPGA BAQ-2.

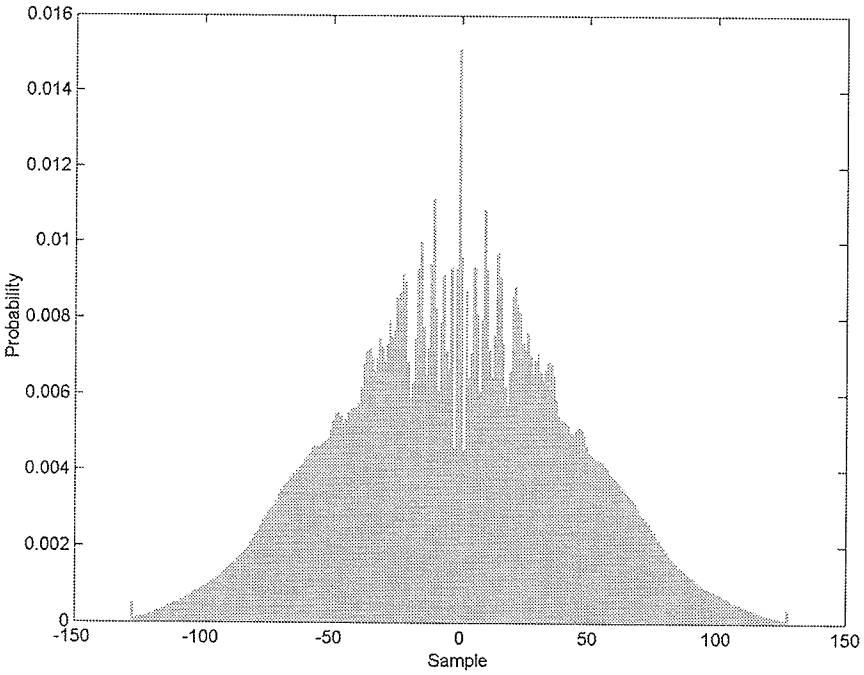


Fig. F.14. Histogram of the E1-25224PS test set quantized with FPGA Wavelet-BAQ-2.

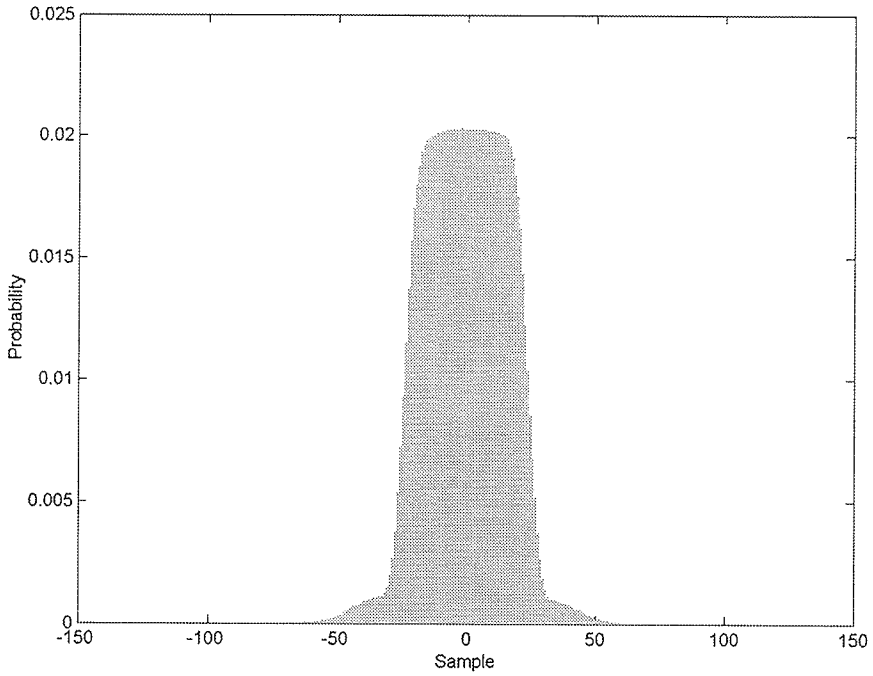


Fig. F.15. Error histogram of the E1-25224PS test set quantized with FPGA BAQ-2.

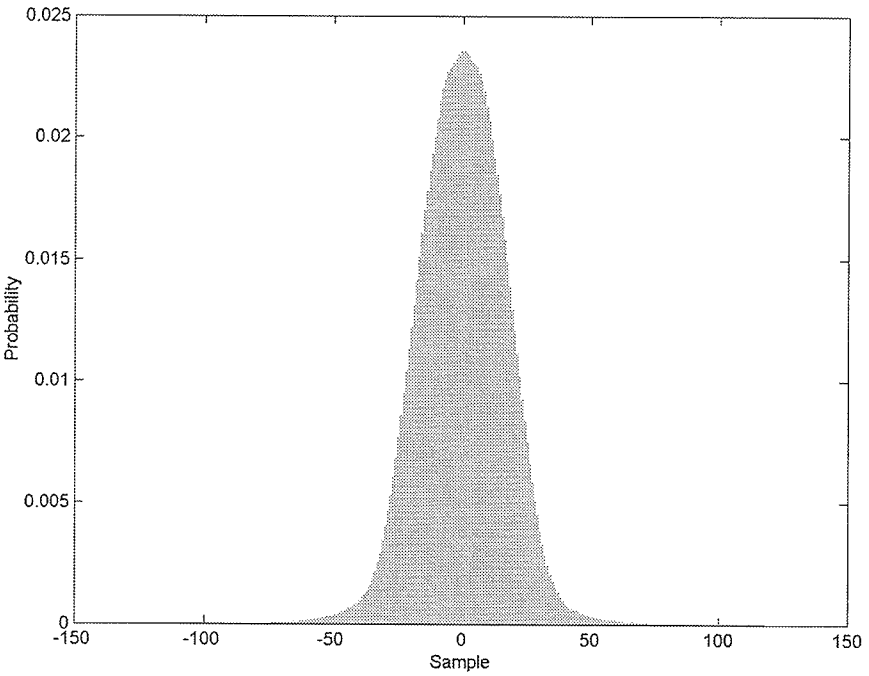


Fig. F.16. Error Histogram of the E1-25224PS test set quantized with FPGA Wavelet-BAQ-2.

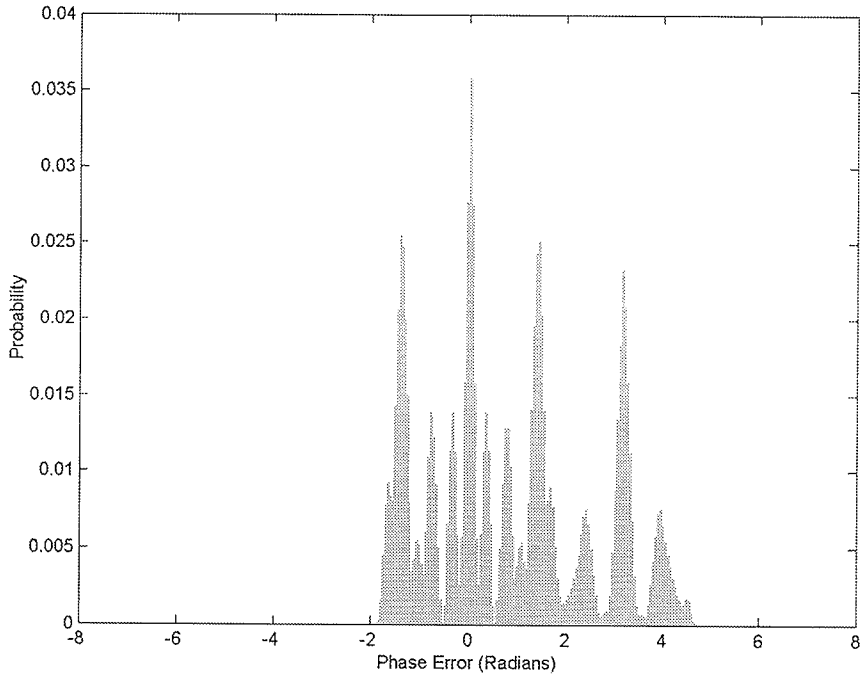


Fig. F.17. Phase error histogram of the E1-25224PS test set quantized with FPGA BAQ-2.

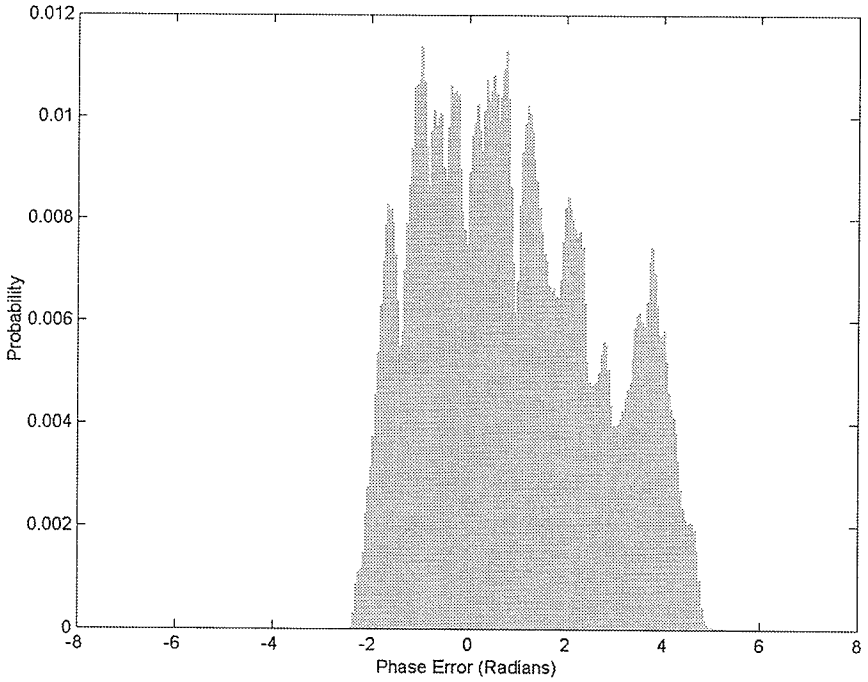


Fig. F.18. Phase error histogram of the E1-25224PS test set quantized with FPGA Wavelet-BAQ-2.

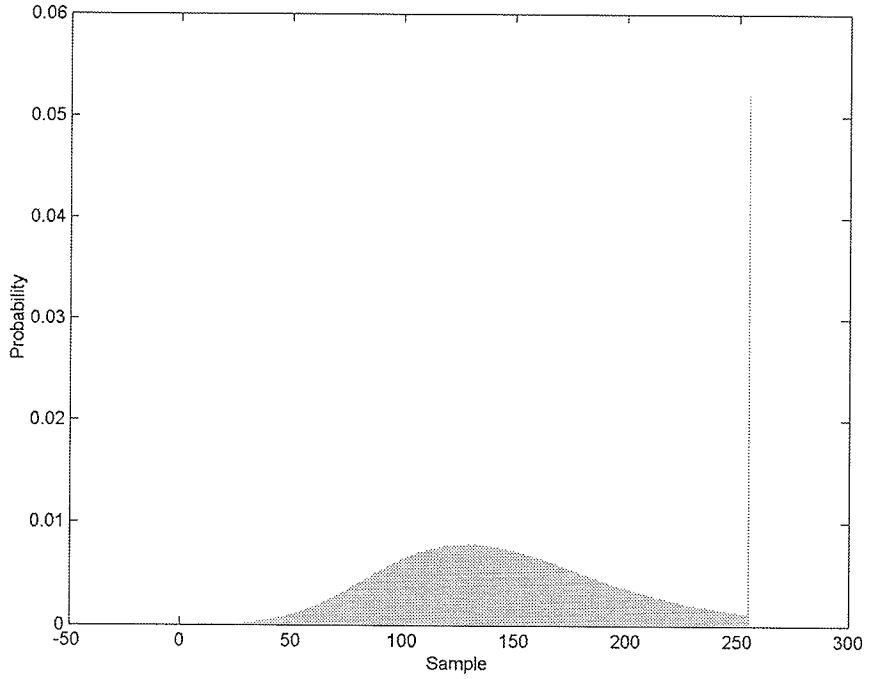


Fig. F.19. Image histogram of the E1-25224PS test set quantized with FPGA BAQ-2.

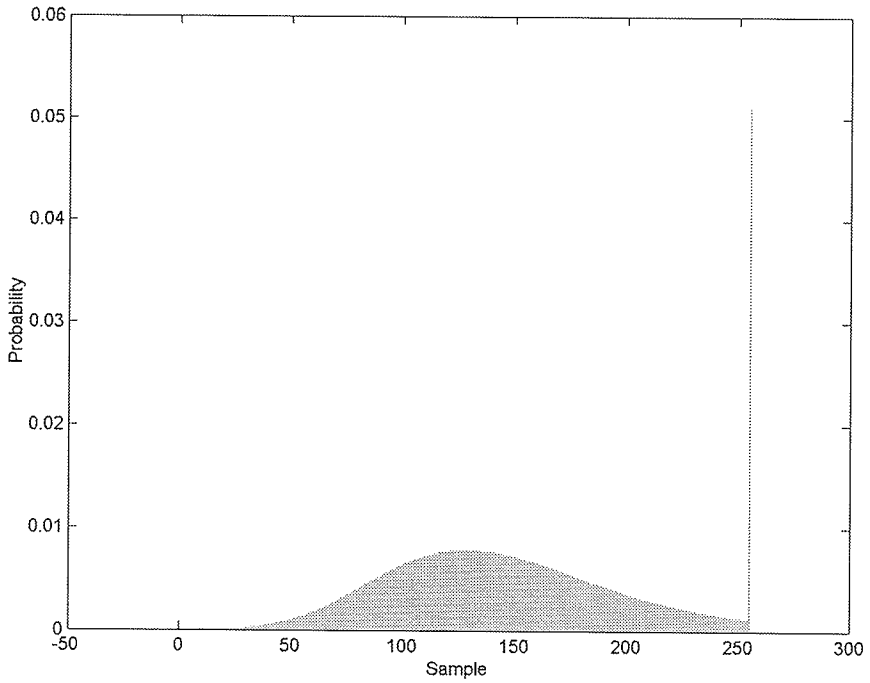


Fig. F.20. Image histogram of the E1-25224PS test set quantized with FPGA Wavelet-BAQ-2.

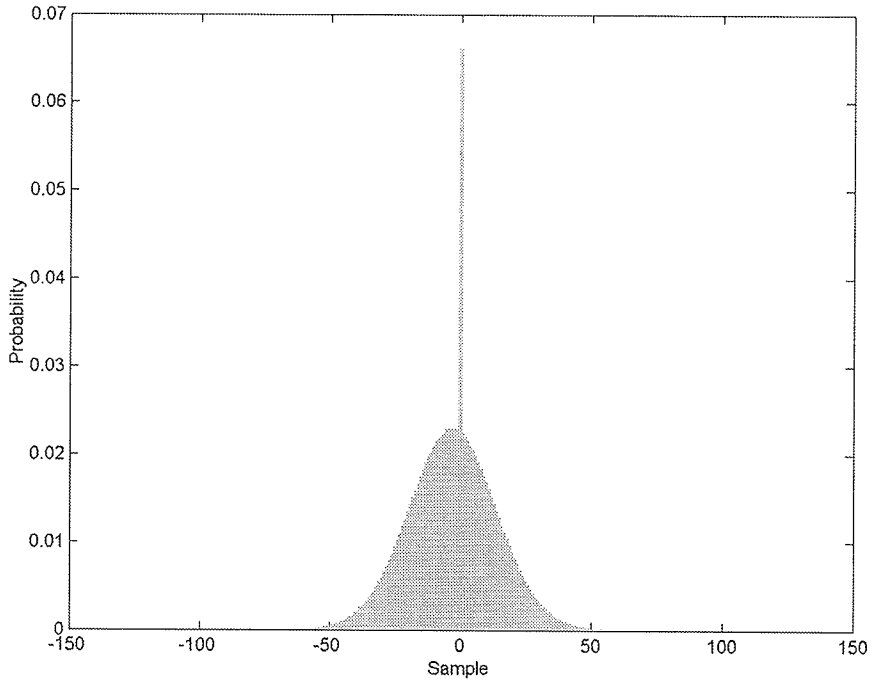


Fig. F.21. Image error histogram of the E1-25224PS test set quantized with FPGA BAQ-2.

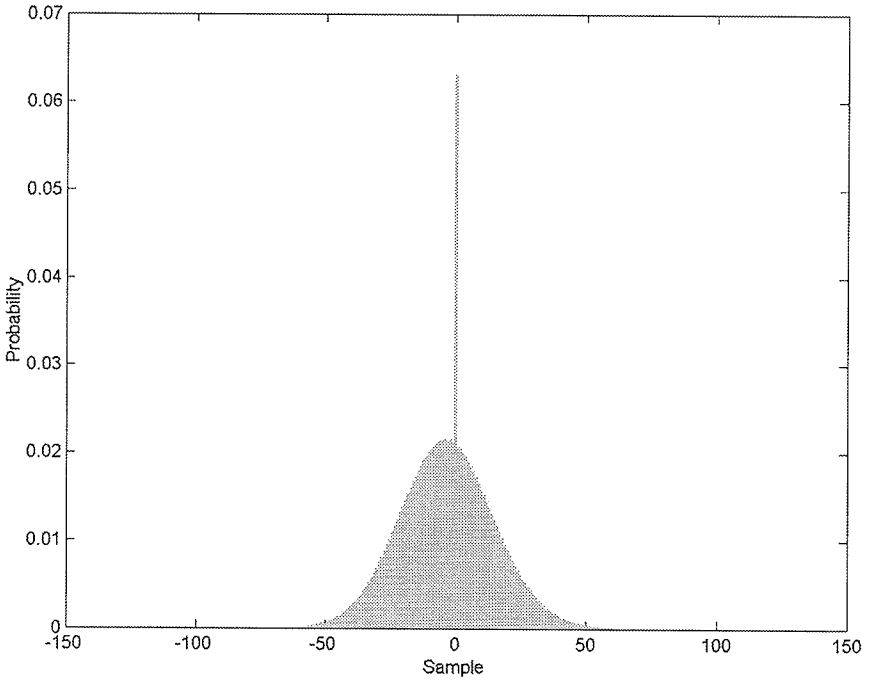


Fig. F.22. Image error histogram of the E1-25224PS test set quantized with FPGA Wavelet-BAQ-2.

F.3 E2-5551PS Test Set Histograms

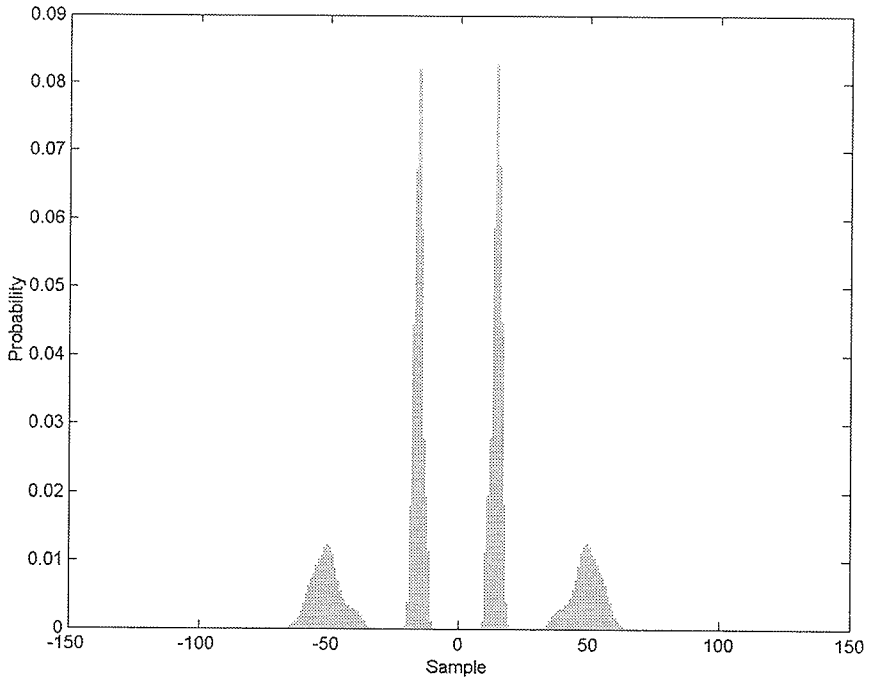


Fig. F.23. Histogram of the E2-5551PS test set quantized with MatLab BAQ-2.

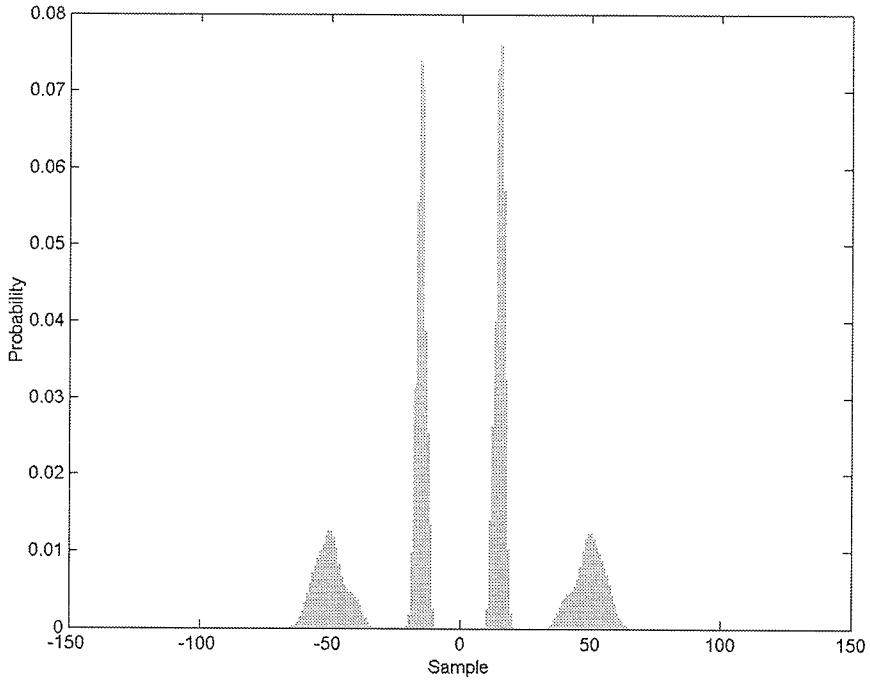


Fig. F.24. Histogram of the E2-5551PS test set quantized with FPGA BAQ-2.

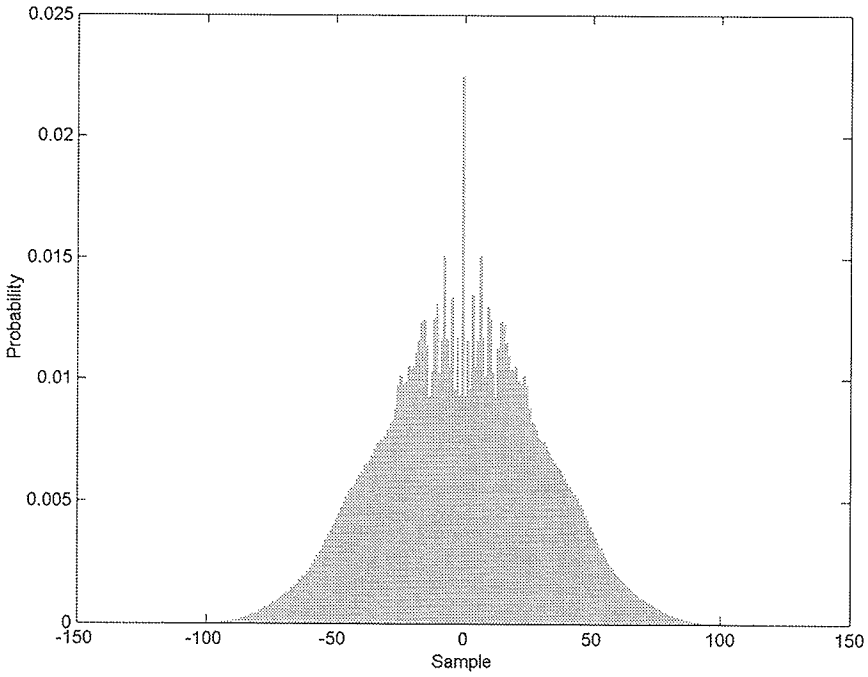


Fig. F.25. Histogram of the E2-5551PS test set quantized with FPGA Wavelet-BAQ-2.

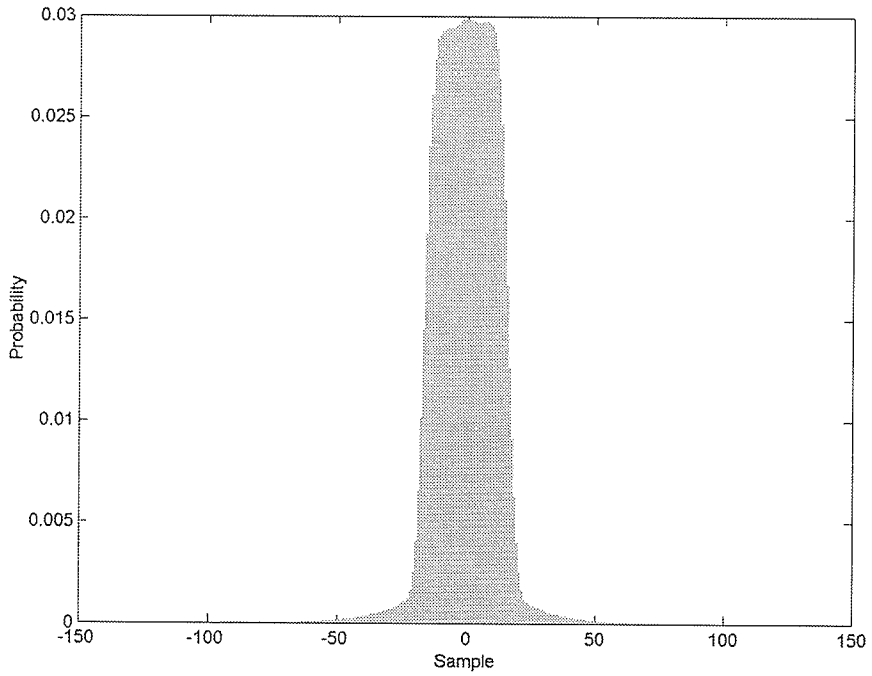


Fig. F.26. Error histogram of the E2-5551PS test set quantized with FPGA BAQ-2.

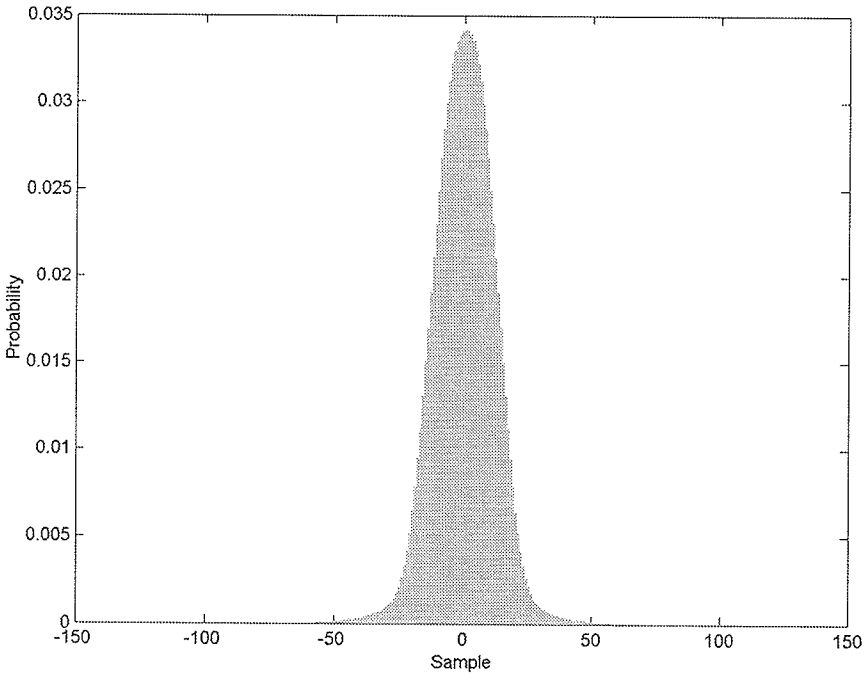


Fig. F.27. Error Histogram of the E2-5551PS test set quantized with FPGA Wavelet-BAQ-2.

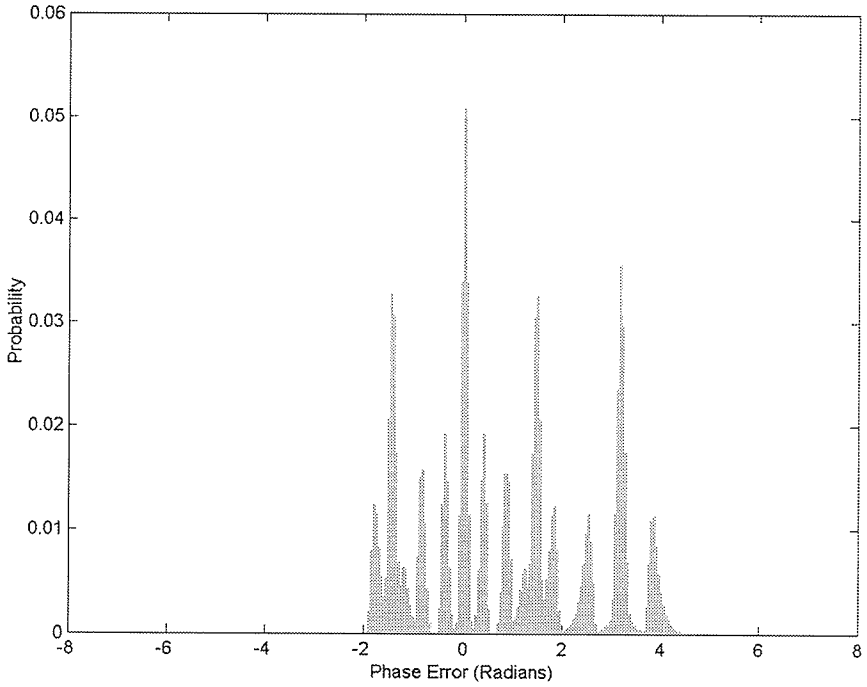


Fig. F.28. Phase error histogram of the E2-5551PS test set quantized with FPGA BAQ-2.

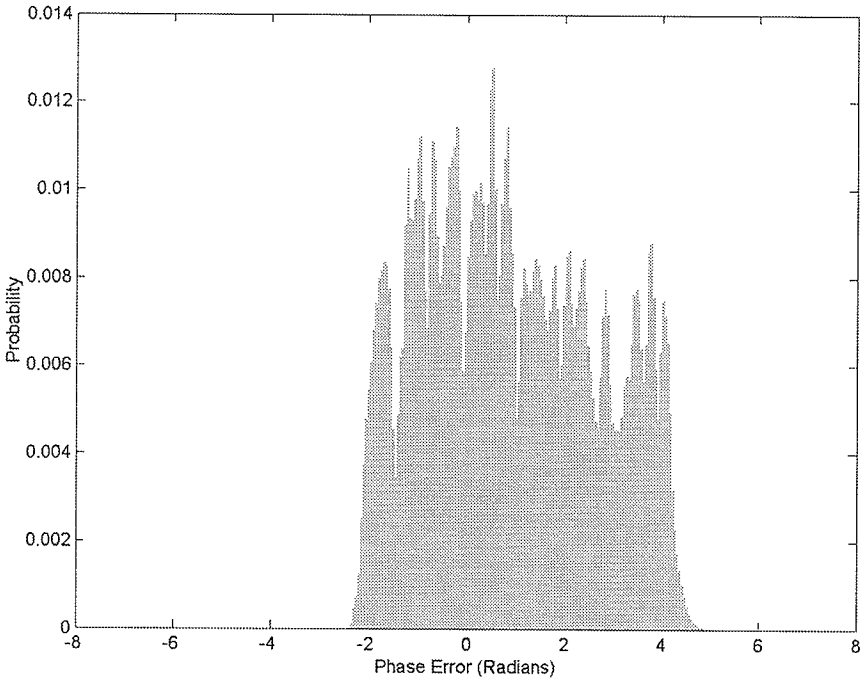


Fig. F.29. Phase error histogram of the E2-5551PS test set quantized with FPGA Wavelet-BAQ-2.

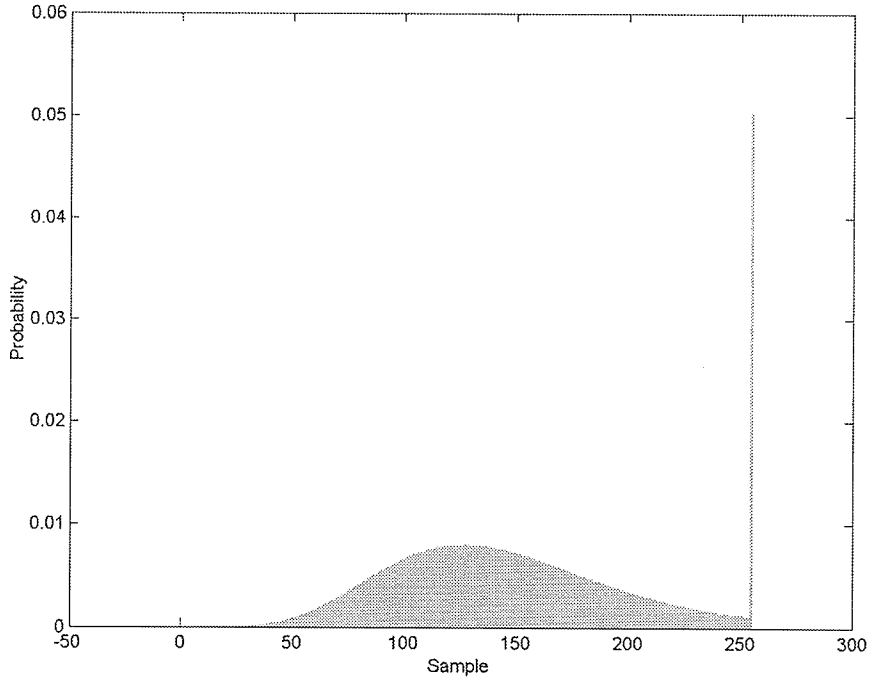


Fig. F.30. Image histogram of the E2-5551PS test set quantized with FPGA BAQ-2.

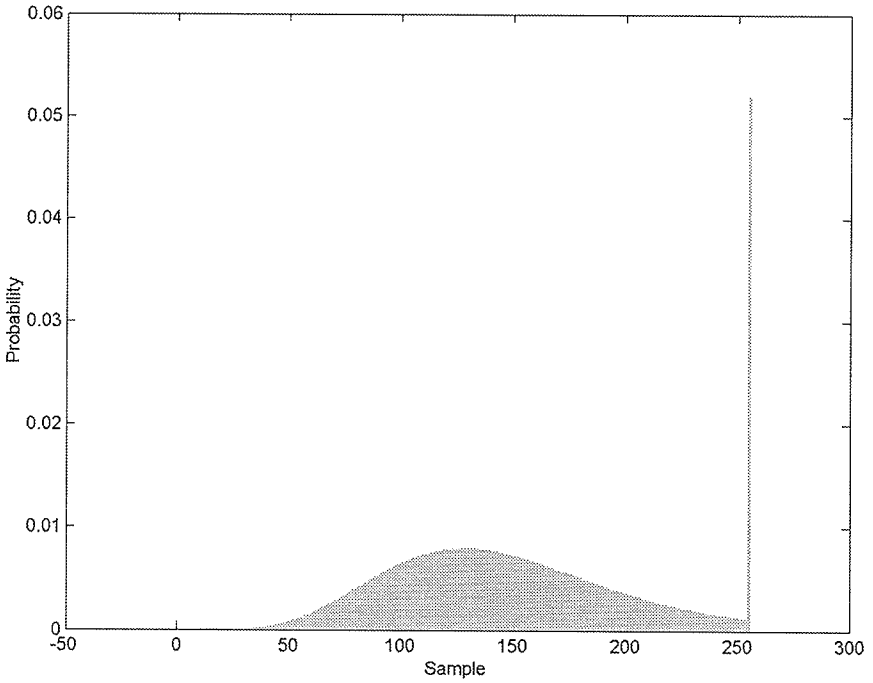


Fig. F.31. Image histogram of the E2-5551PS test set quantized with FPGA Wavelet-BAQ-2.

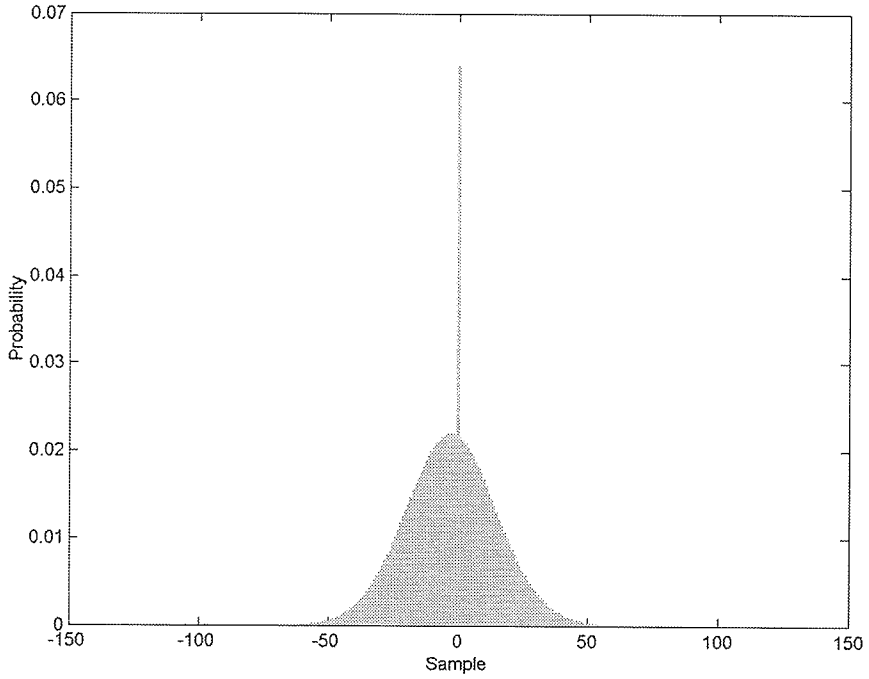


Fig. F.32. Image error histogram of the E2-5551PS test set quantized with FPGA BAQ-2.

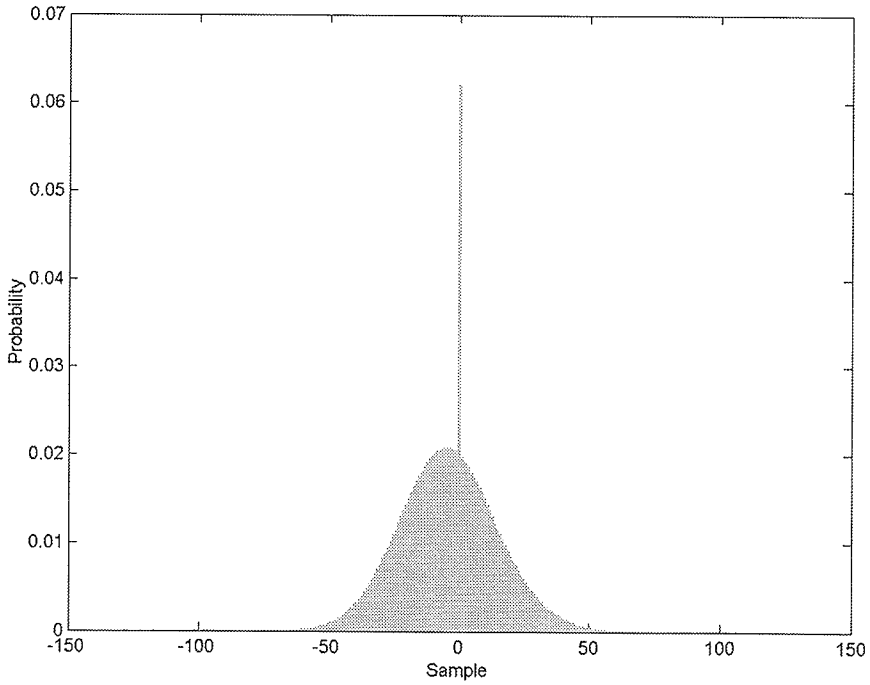


Fig. F.33. Image error histogram of the E2-5551PS test set quantized with FPGA Wavelet-BAQ-2.

F.4 R1-24576PS Test Set Histograms

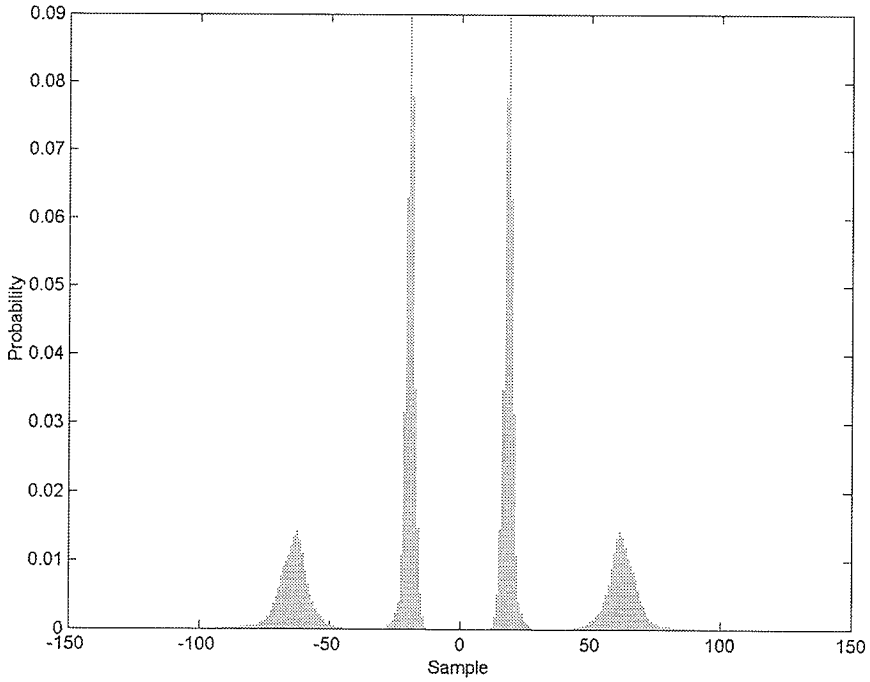


Fig. F.34. Histogram of the R1-24576PS test set quantized with MatLab BAQ-2.

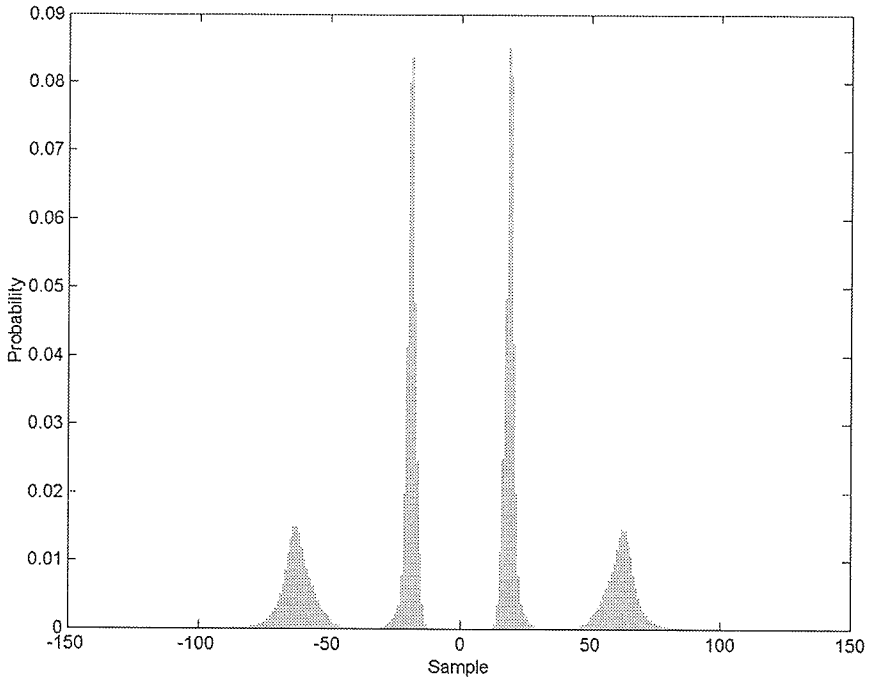


Fig. F.35. Histogram of the R1-24576PS test set quantized with FPGA BAQ-2.

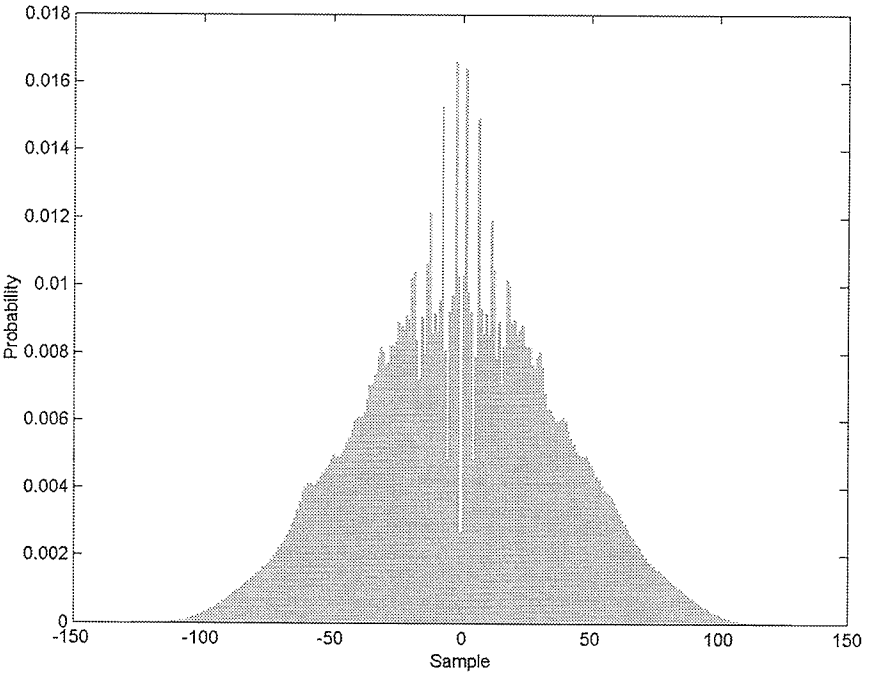


Fig. F.36. Histogram of the R1-24576PS test set quantized with FPGA Wavelet-BAQ-2.

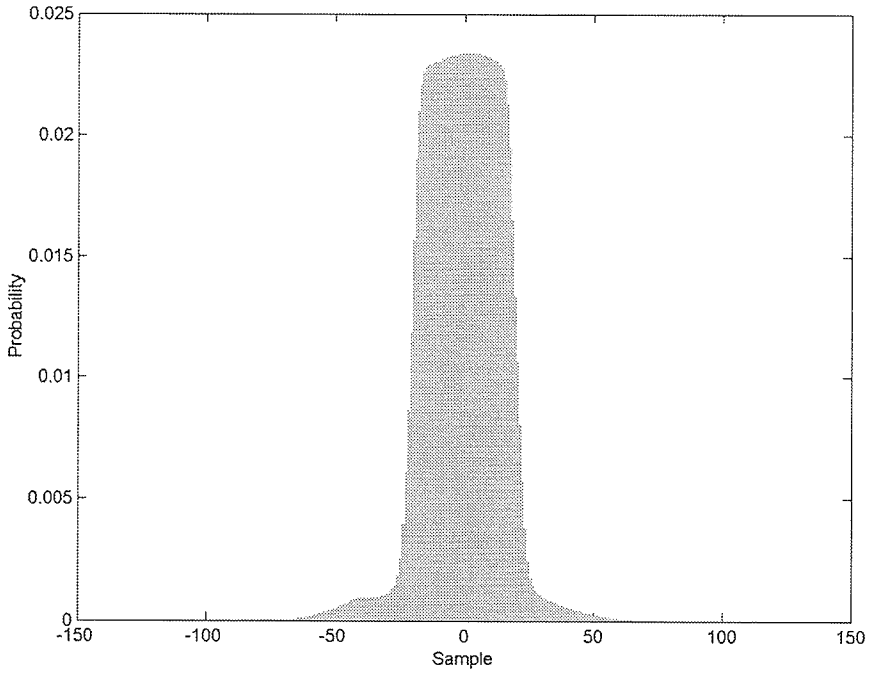


Fig. F.37. Error histogram of the R1-24576PS test set quantized with FPGA BAQ-2.

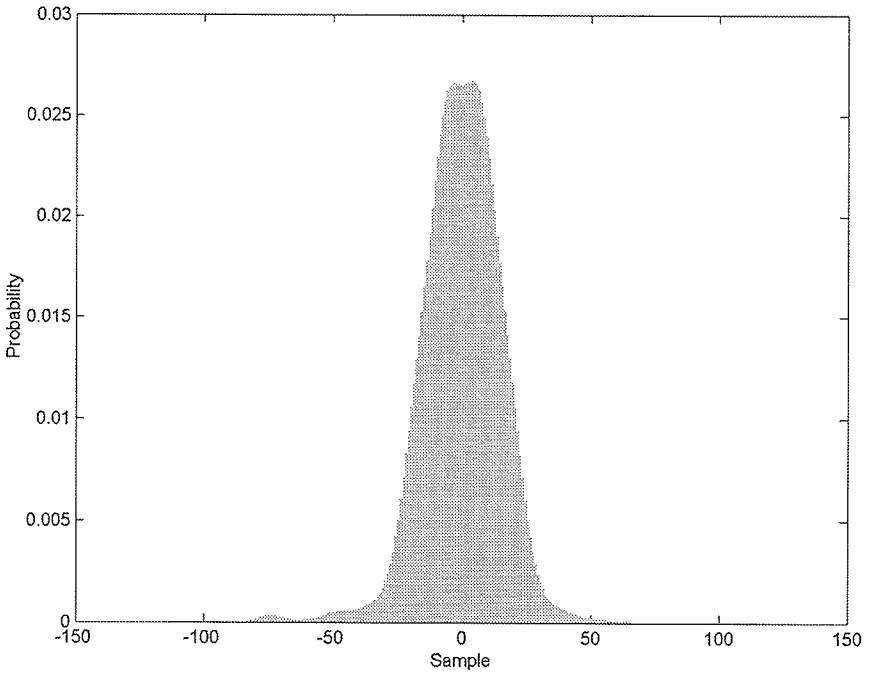


Fig. F.38. Error Histogram of the R1-24576PS test set quantized with FPGA Wavelet-BAQ-2.

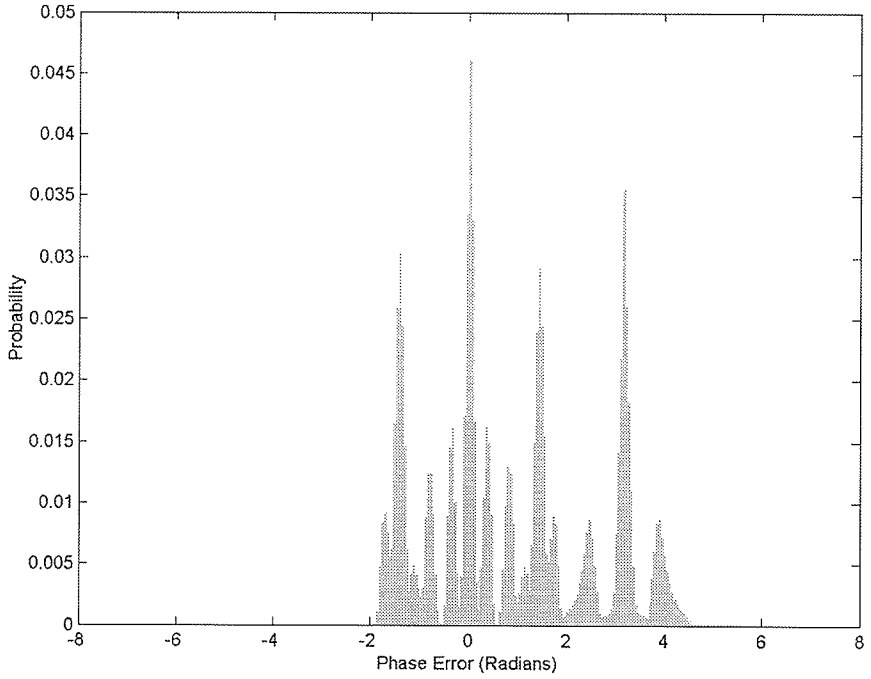


Fig. F.39. Phase error histogram of the R1-24576PS test set quantized with FPGA BAQ-2.

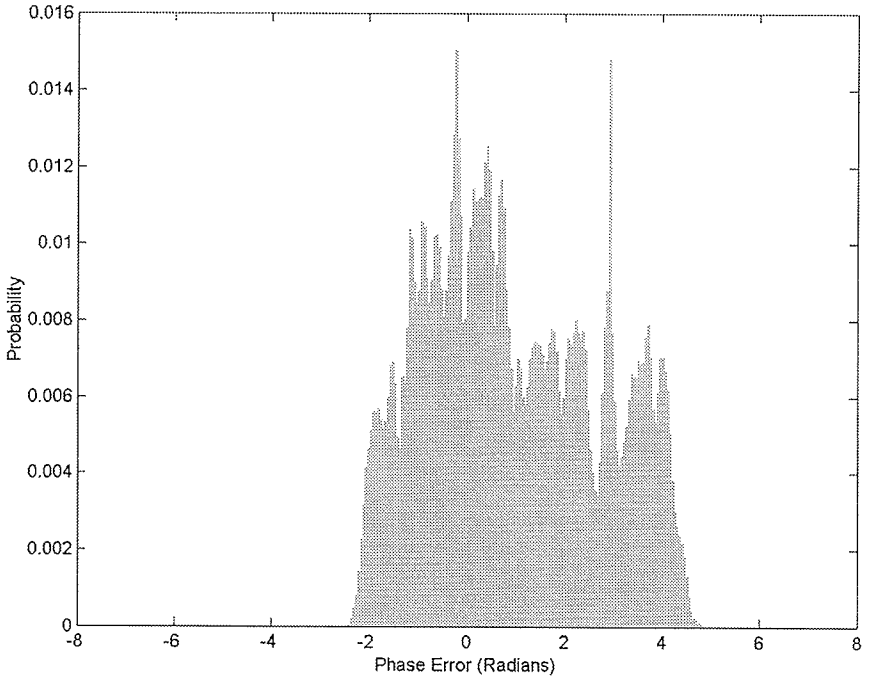


Fig. F.40. Phase error histogram of the R1-24576PS test set quantized with FPGA Wavelet-BAQ-2.

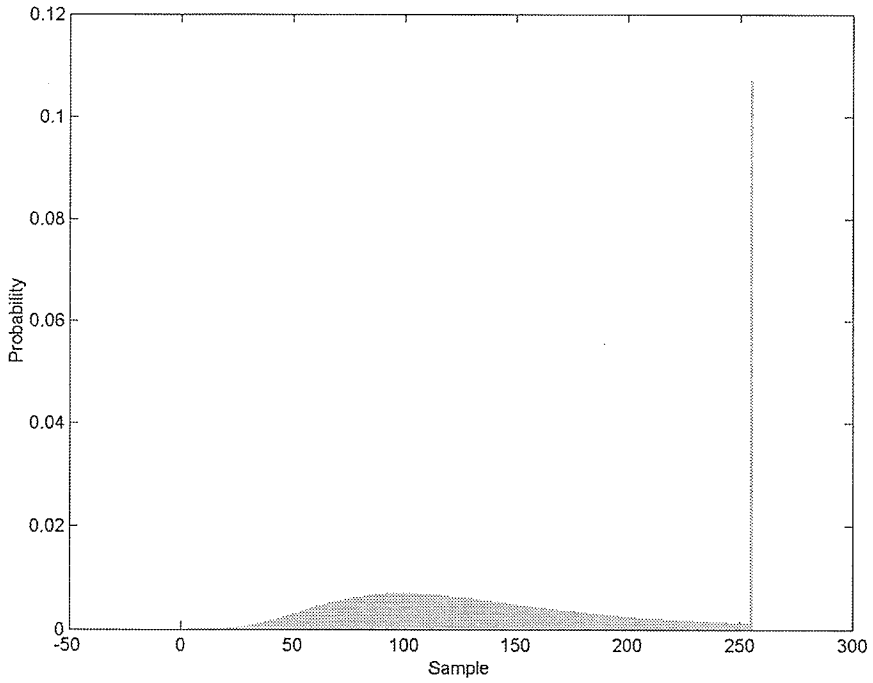


Fig. F.41. Image histogram of the R1-24576PS test set quantized with FPGA BAQ-2.

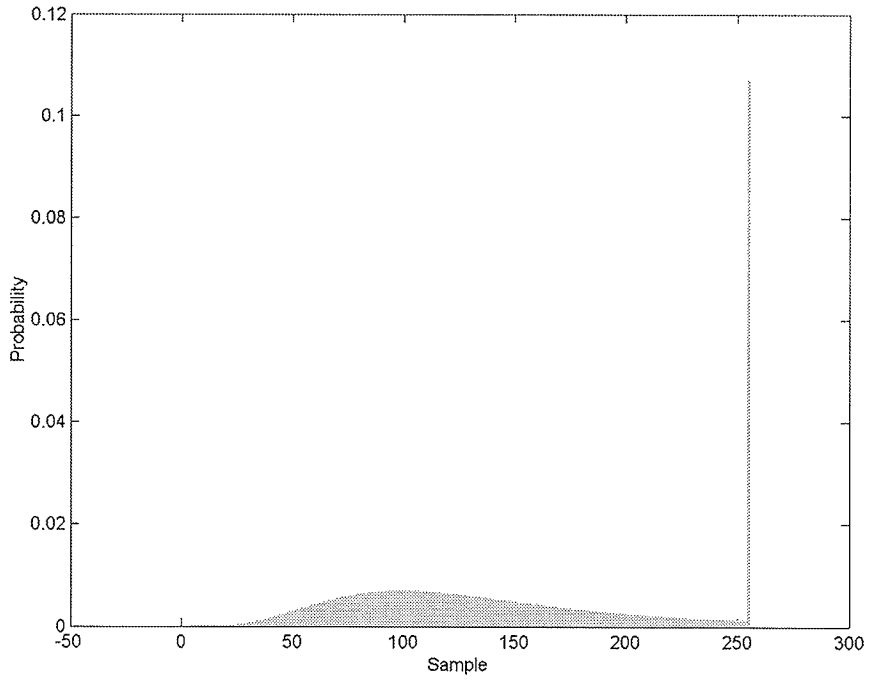


Fig. F.42. Image histogram of the R1-24576PS test set quantized with FPGA Wavelet-BAQ-2.

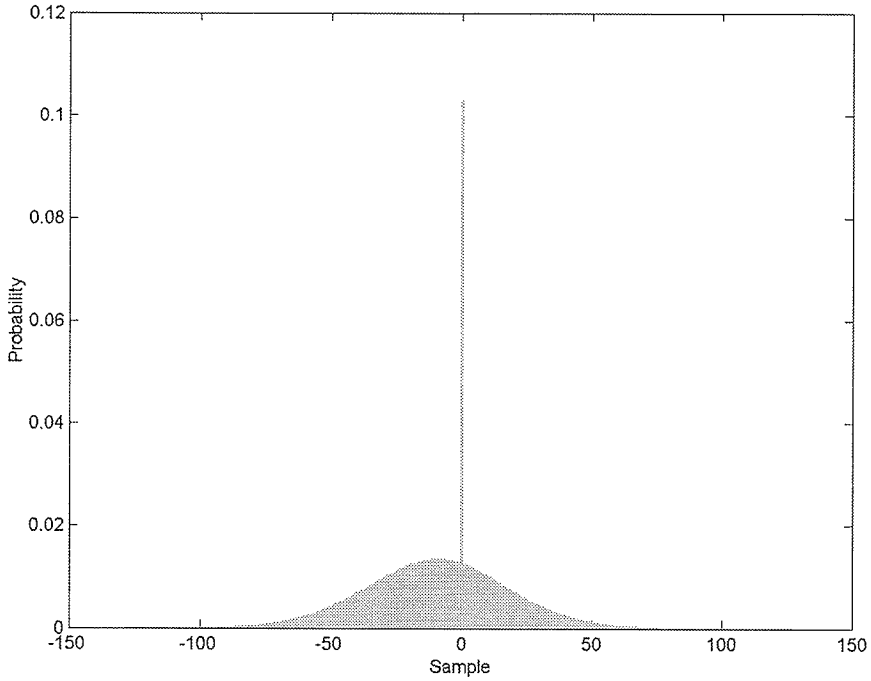


Fig. F.43. Image error histogram of the R1-24576PS test set quantized with FPGA BAQ-2.

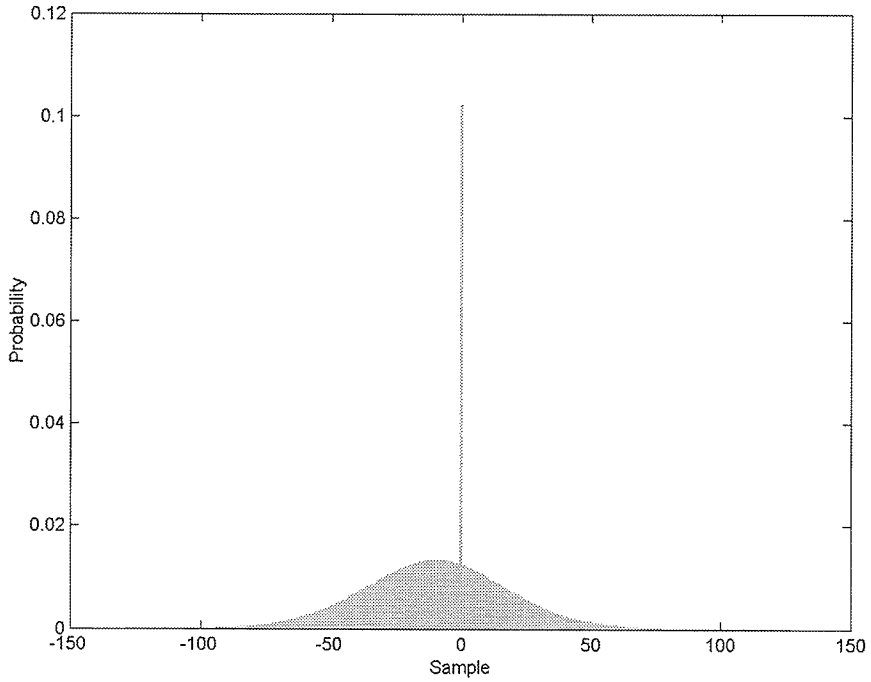


Fig. F.44. Image error histogram of the R1-24576PS test set quantized with FPGA Wavelet-BAQ-2.

F.5 R1-24919PS Test Set Histograms

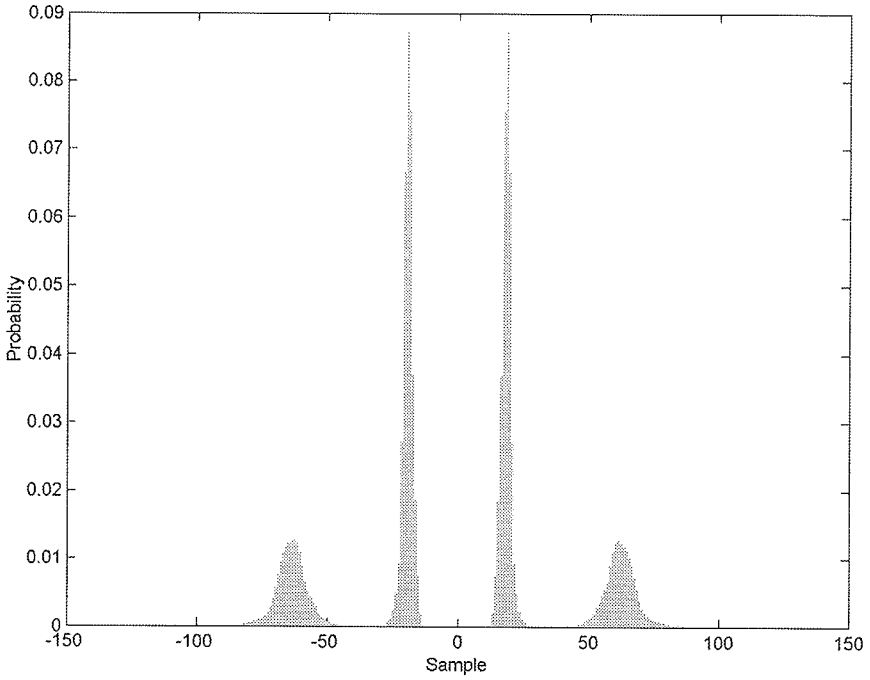


Fig. F.45. Histogram of the R1-24919PS test set quantized with MatLab BAQ-2.

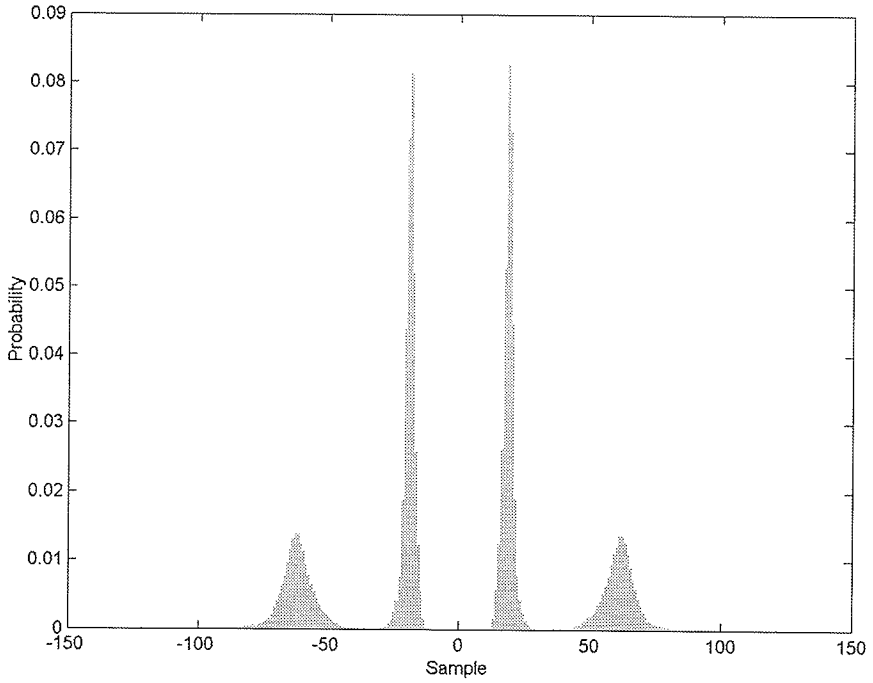


Fig. F.46. Histogram of the R1-24919PS test set quantized with FPGA BAQ-2.

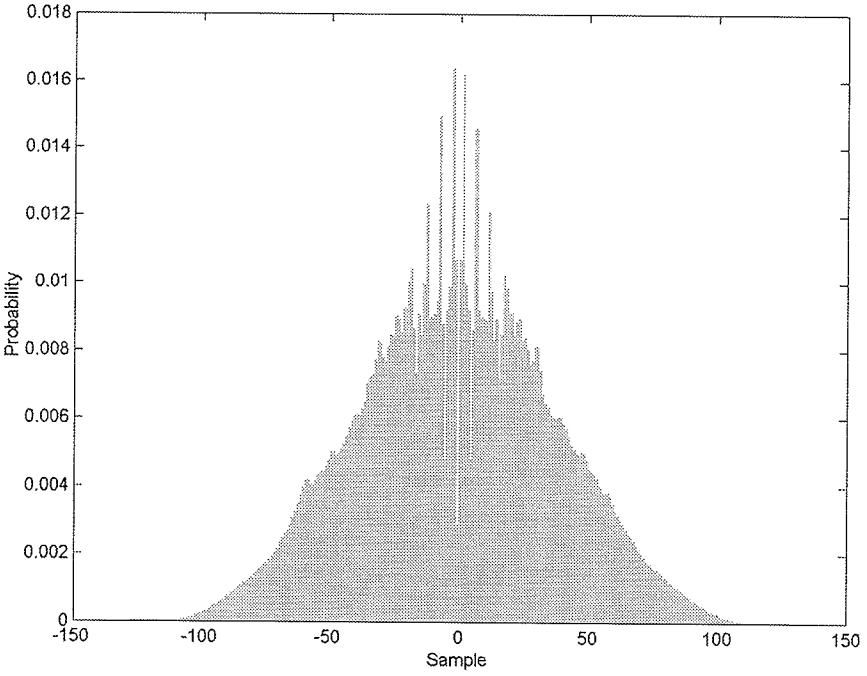


Fig. F.47. Histogram of the R1-24919PS test set quantized with FPGA Wavelet-BAQ-2.

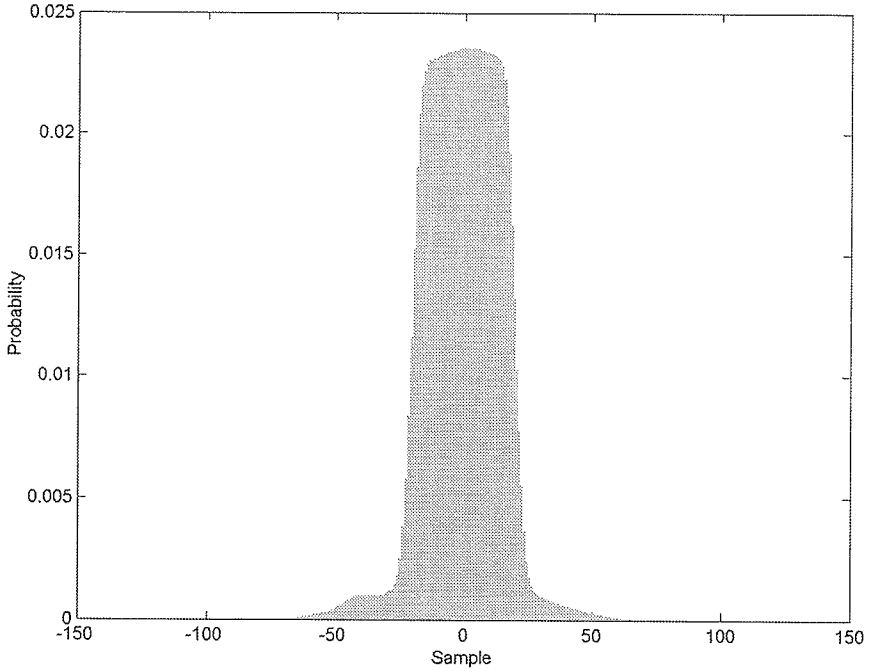


Fig. F.48. Error histogram of the R1-24919PS test set quantized with FPGA BAQ-2.

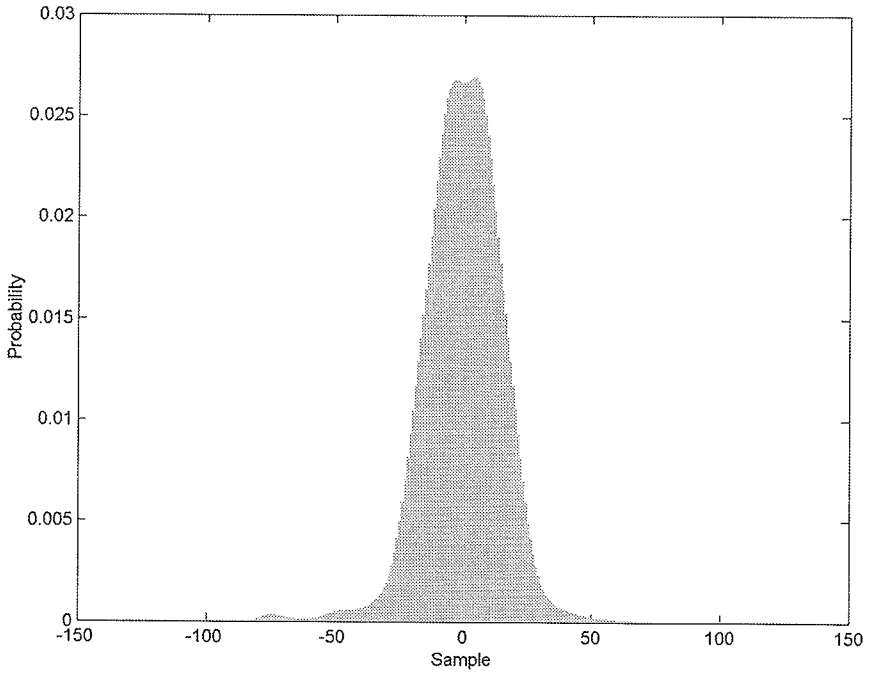


Fig. F.49. Error Histogram of the R1-24919PS test set quantized with FPGA Wavelet-BAQ-2.

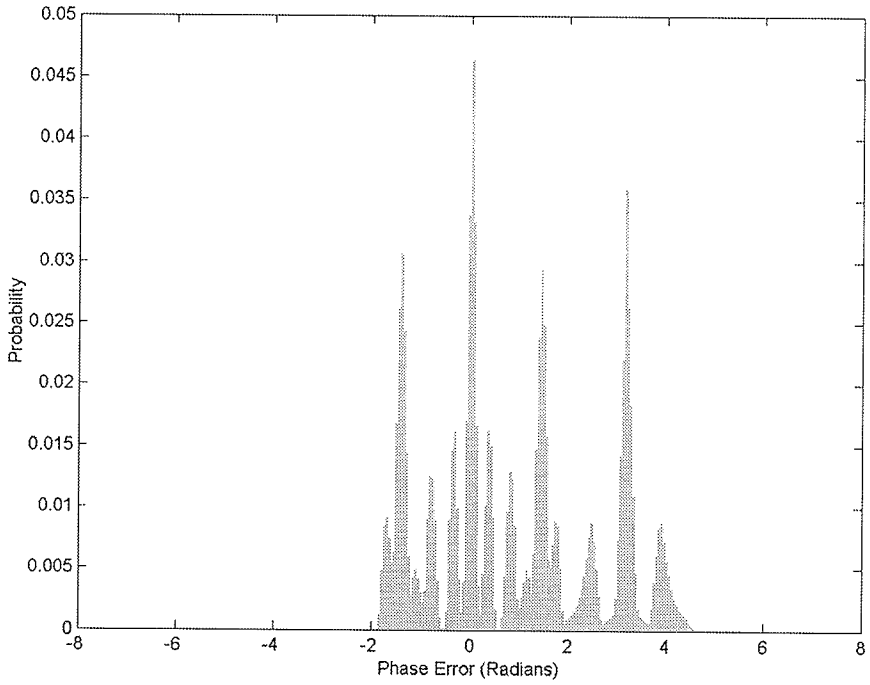


Fig. F.50. Phase error histogram of the R1-24919PS test set quantized with FPGA BAQ-2.

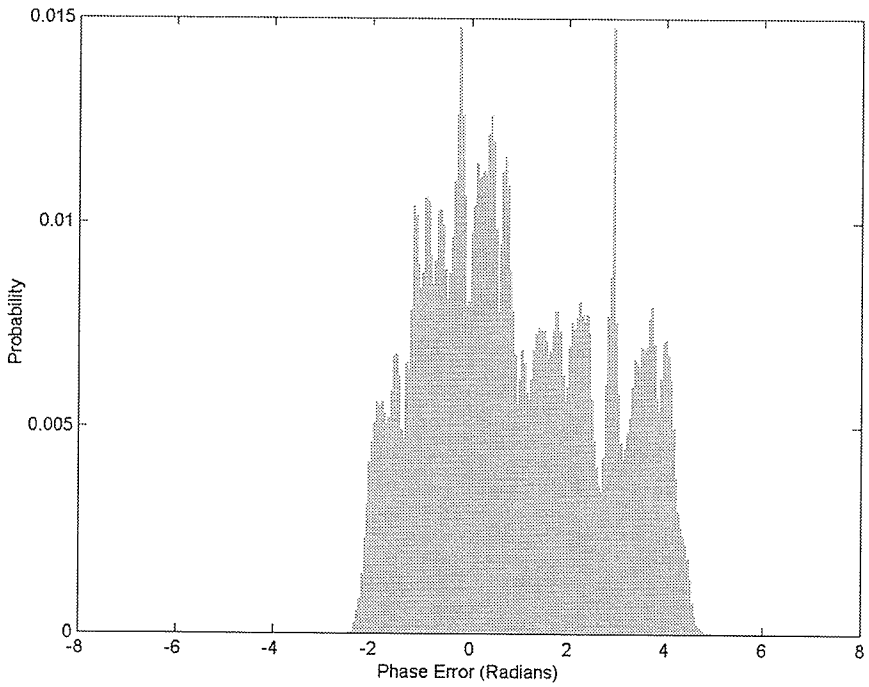


Fig. F.51. Phase error histogram of the R1-24919PS test set quantized with FPGA Wavelet-BAQ-2.

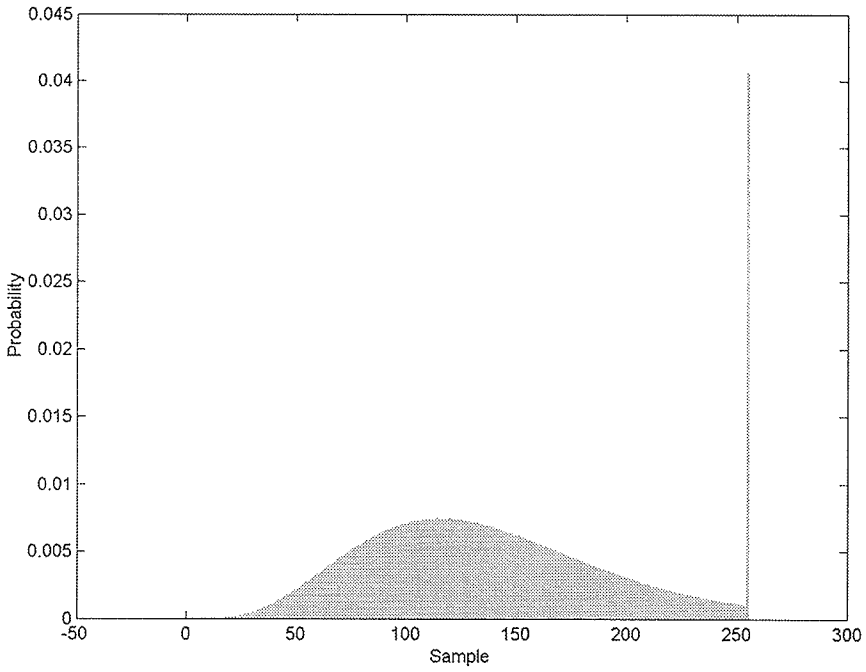


Fig. F.52. Image histogram of the R1-24919PS test set quantized with FPGA BAQ-2.

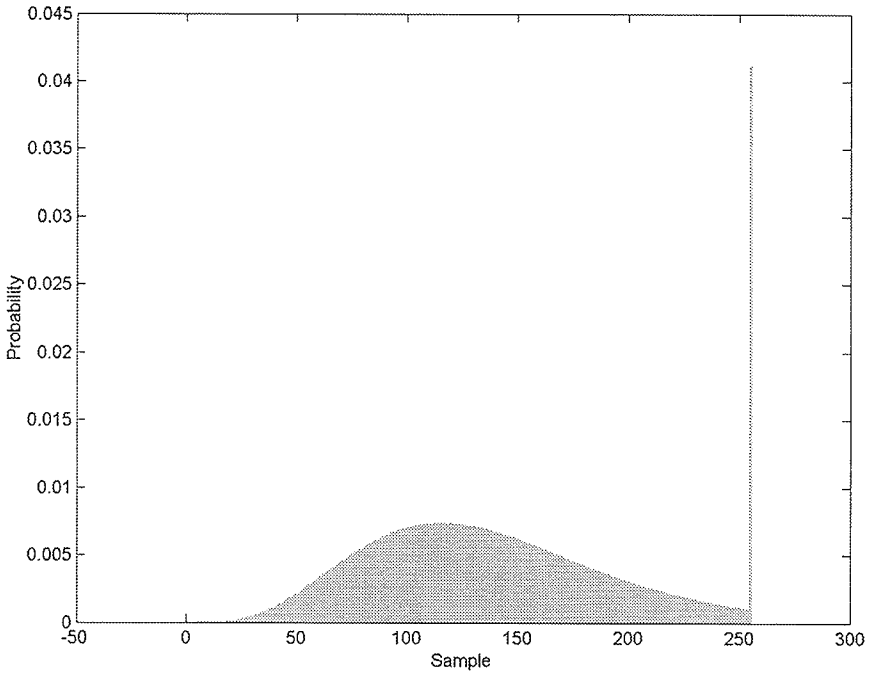


Fig. F.53. Image histogram of the R1-24919PS test set quantized with FPGA Wavelet-BAQ-2.

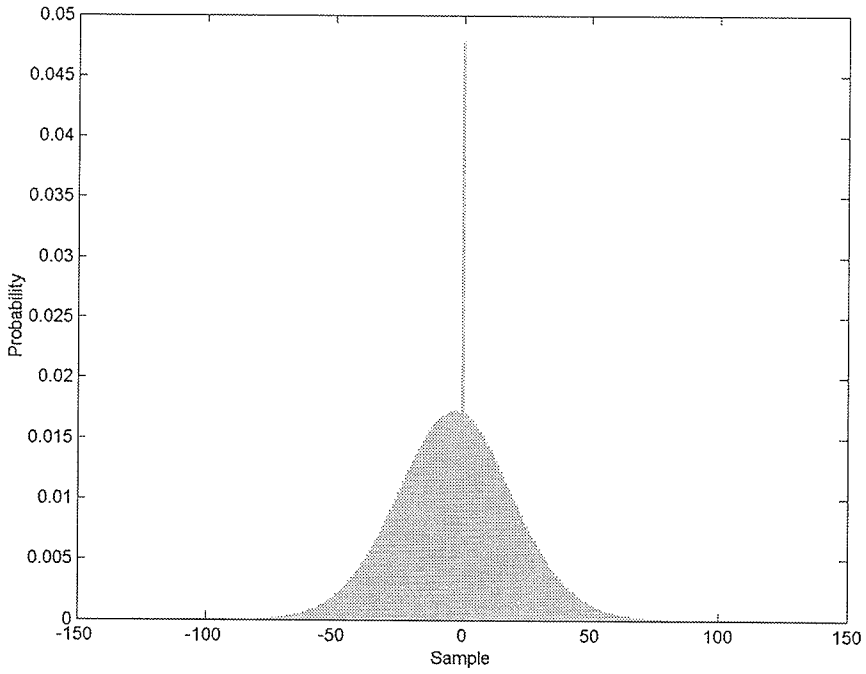


Fig. F.54. Image error histogram of the R1-24919PS test set quantized with FPGA BAQ-2.

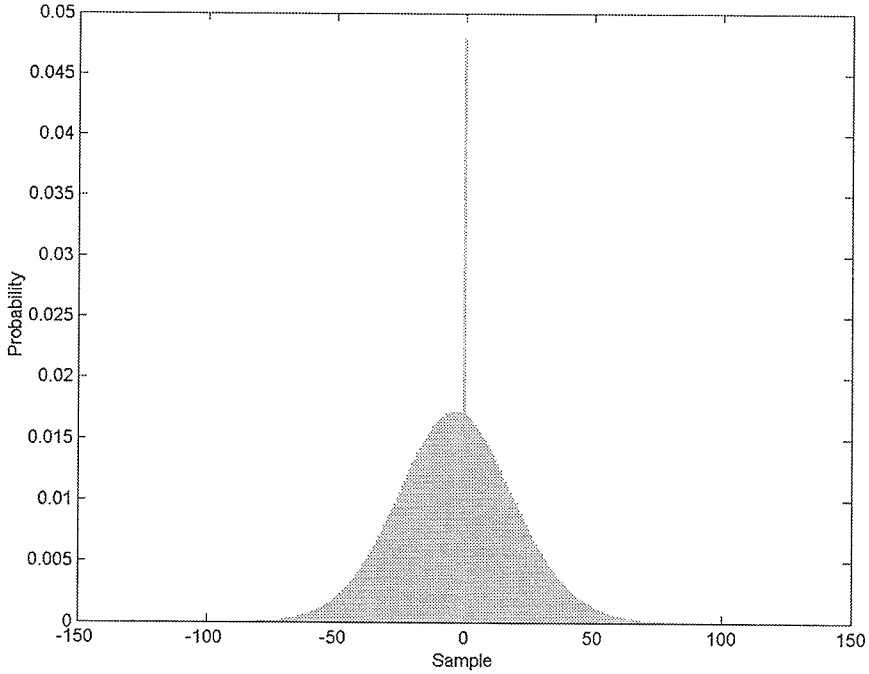


Fig. F.55. Image error histogram of the R1-24919PS test set quantized with FPGA Wavelet-BAQ-2.