Echo Synthesis for Decoy Application

A dissertation
submitted in partial fulfillment of the requirements
for the degree of

Master of Technology

By

Sukhendu Sharma
(Roll No. 03307416)

under the supervision of

Prof. P. C. Pandey

Department of Electrical Engineering
Indian Institute of Technology, Bombay
July 2005

INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY

M. Tech. Dissertation Approval

Dissertation entitled, “Echo Synthesis for Decoy Application”, submitted by Sukhendu Sharma (Roll No. 03307416) is approved for the award of degree of Master of Technology in Electrical Engineering with specialization in Communication Engineering.

Supervisor : _____ (Prof. P. C. Pandey)

Internal examiner : _____ (Prof. V. M. Gadre)

External examiner : _____ (Dr. V. K. Madan)

Chairman : _____ (Prof. K. Moudgalya)

Date: July 7, 2005

Sukhendu Sharma / Prof. P. C. Pandey (supervisor): "Echo synthesis for decoy application," *M. Tech dissertation*, Department of Electrical Engineering, Indian Institute of Technology, Bombay, July 2005.

Abstract

During an electronic warfare in sea, a torpedo is fired from a ship (or submarine) to attack and destroy its enemy vessel. Once a torpedo is identified by the sonar of the vessel, the vessel launches a small expendable electronic device, called decoy, to protect itself and employs an escape maneuver. The aim of the project is to develop a technique for synthesizing and transmitting a signal by the decoy that can be interpreted by the torpedo sonar receiver as the reflection of its own transmitted signal from the target vessel. The echo synthesis should take care of various modulation schemes employed in the torpedo sonar and should have delay, Doppler-shift, and echo elongation corresponding to the distance, speed, and size of the moving target vessel. The approach adopted in this project is that, first the decoy detects the incoming signal pulse transmitted by the torpedo, and then retransmits the received signal by introducing in it the appropriate features without analyzing the input signal. The complete algorithm of echo synthesis has been implemented using MATLAB, and tested with different types of signal in additive white noise. The approach of echo synthesis by considering the entire signal pulse received by the decoy requires large computation and storage capabilities of the hardware to implement the algorithms. Hence, a second approach has been investigated in which the echo synthesis is carried out by using overlap-add processing of the input sequence. Finally, the individual modules of the echo synthesis process, for the overlap-add processing, have been implemented in real-time for a DSP hardware based on ADSP-21160 processor.

ACKNOWLEDGEMENTS

I would like to express my deep sense of gratitude to my supervisor, Prof. P. C. Pandey, for his invaluable guidance, support and encouragement throughout the course of this project.

I also thank the Director, Naval Science & Technological Lab. Visakhapatnam, DRDO, Ministry of Defence, Govt. of India, for granting me permission for the M.Tech. programme at IIT, Bombay under the R & T scheme of DRDO.

I would also like to thank all lab mates and friends for all the assistance provided.

IIT, Bombay Sukhendu Sharma

July 2005

CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Overview

During a sea warfare, a torpedo is used as a weapon to destroy the enemy vessels. Warship with its large size, poor maneuverability, and low speed is vulnerable against an attack by a torpedo that moves at a speed almost three times that of the ship [1]. A torpedo employs a sonar for homing on the target ship. To protect itself, the target vessel under attack, should employ an underwater electronic counter-measure (ECM) technique. In the ECM technique, the sonar of the target vessel identifies that a torpedo is fired and launches a small electronic device, called decoy, and employs an escape maneuver. The decoy being a small underwater, stationary, and expendable electronic device should make itself appear like a real target (a ship) so that the attacking torpedo will home on to the decoy instead of the ship.

The torpedoes, which are currently under use, work in passive as well as active mode. In passive mode, the torpedo does not transmit any signal but listens to the noise like signature generated by the ship to detect and identify the target. In active mode, the torpedo transmits an ultrasonic burst and analyzes the reflected signal to determine the presence of a target, its range and velocity. The process of transmitting a known signal and listening for its reflections is referred to generally as Range-Doppler detection. Although the actual underwater detection schemes employ both passive and active modes of operation, the scope of this project is restrained to the development of the decoy algorithms in the case of active mode of operation of the torpedo. The decoy electronics should receive the ultrasonic signal from the torpedo sonar and transmit an ultrasonic signal that can be interpreted, by the torpedo sonar receiver, as reflected from the target vessel. In the active mode of operation, the choices of the band-width, frequency and the type of the signal used by the torpedo are unknown, and hence the synthesis of the signal

in the decoy should take care of various modulation schemes that are employed in the torpedo.

One way the decoy may function is by detecting the incoming signal pulse and analyzing the signal for its various parameters like carrier frequency, type of modulation, and modulation parameters thereby synthesizing a signal to be transmitted, simulating the torpedo signal echoed from the ship. A method similar to the principle of linear least square analysis has been adopted for estimating the frequency of a noisy sinusoid [2]. Based on this theoretical evaluation of the estimator, the problem of the parameter estimation of chirp signals has been addressed in [3]. Basically, the phase of the observed sequence is modeled as a polynomial embedded in white noise, which implies that, first a phase unwrapping is accomplished, and then linear regression techniques are applied to obtain the estimates of the parameters. Although the estimators are accurate and simple for on-line implementation, the signal-to-noise ratio required is quite high, almost 10 dB and more for the polynomial of degree 2. Increasing the degree also increases the required SNR for proper performance. In this project, the approach adopted is to detect the incoming signal pulse and retransmit the received burst with features such as a delay, Doppler shift, and echo extension corresponding to the ship distance, speed, and size without analyzing the input signal.

One way to model the reflection from a moving target is by imparting a frequency shift to the reflected signal. If the entire band of the signal is shifted laterally by an amount equal to the Doppler change of the center frequency of the original band, then the method is called narrowband processing where the bandwidth of the signal remains same. In wideband processing, each frequency component of the original signal is shifted by an amount equal to the Doppler change corresponding to that particular frequency thereby changing the bandwidth of the signal. It can be shown [11] that the narrowband processing constraints the time-bandwidth product such that

$$TB << \frac{c}{2v} \tag{1.1}$$

where c = velocity of sound in water, and v = the relative velocity between source and target. Considering a pulse-width T = 100 ms, bandwidth B = 2 kHz, and c = 1500 m/s, the relation given in Eqn. 1.1 holds for $v << 3.75$ m/s. Since the velocity of the ship can be about 10 m/s (20 knots), we can not use narrowband processing.

1.2 Project objective

The objective of the project is to develop a technique of echo generation for use in the decoy to simulate a target vessel. The echo synthesis is to be carried out without analyzing the incoming signal in terms of its bandwidth and the type of modulation being used. Being non-specific to the type of modulation, the method should work for all types of modulation and will not be affected by the errors in the estimation of modulation parameters.

In this project, two approaches have been adopted to implement the echo synthesis in the decoy. In the first approach, the echo synthesis has been carried out considering the entire input signal detected by the decoy. However, this approach of echo synthesis requires large computation and storage capabilities of the hardware. Hence, another approach of synthesizing the echo by processing smaller segments of the input signal has also been investigated. The complete algorithm of signal detection and echo synthesis in the case of an additive white Gaussian background noise has been developed in MATLAB with simulated signals. Finally, the entire algorithm has been implemented for real-time operation in a DSP hardware based on the ADSP-21160 digital signal processor.

1.3 Report outline

Chapter 2 gives the background of the problem along with a description of the decoy and torpedo operation. Chapter 3 describes the techniques to implement the complete echo generation in the decoy. The techniques include detection of the input signal in noise, SNR improvement of the received signal, implementing Doppler shift and subsequently the echo extension of the signal. Chapter 4 provides the implementation techniques as well as the simulation results of implementing the complete echo generation algorithms in MATLAB. Chapter 5 presents the real-time implementation of the decoy algorithms, through software, for a hardware based on ADSP-21160 digital signal processor. Chapter 6 summarizes the work done in this project.

Chapter 2

TORPEDO AND DECOY OPERATIONS

The target vessel launches decoy at the time when the torpedo is reasonably close to the vessel (around 2 km) and employs an escape maneuver. Typical speeds of torpedoes would be 17.5-25 m/s (35-50 knots)* and the escape speed of a ship will be around 10 m/s (20 knots). The operating frequencies of a torpedo lie in the range between 17 and 80 kHz with the type of the signal used for underwater detection, in active mode, as [1], [4], [5]

a) Pulsed continuous wave (PCW), a single frequency sinusoidal wave

$$s_{PCW}(t) = \sin(2\pi f t) \tag{2.1}$$

for a certain duration T.

b) Pulsed linear frequency modulated (PLFM) wave, a sinusoidal wave whose frequency increases linearly with time for a certain duration. A LFM signal with a start frequency f_{st}, a pulse duration T, and a frequency sweep δf, is given as

$$s_{LFM}(t) = \sin[2\pi(f_{st}t + \frac{1}{2}\frac{\delta f}{T}t^2)] \tag{2.2}$$

c) Pulsed coded frequency sequence (CFS) wave, a signal of duration T, formed by concatenating in time a number of different single-frequency sinusoidal waves, each of an equal time duration.

In the active mode, the torpedo transmits a burst of signal of width ranging from 5 ms to 100 ms with various pulse repetition intervals (PRIs) of 2.7 s, 1.35 s and 0.65 s. The decoy electronics will acquire the burst of signal transmitted by the torpedo, detect the burst and retransmit a suitable ship-like signature that will fool the incoming torpedo to home on to the decoy instead of the target vessel, thus protecting the vessel.

* 1 knot ≈ 0.5 m/s

2.1 Decoy: Principle of operation

The decoy is a small, expendable, cylindrical, underwater, electronic object with separate omni-directional transducers used for reception and transmission of acoustic signals. The sequence of events that takes place in the decoy, after it is launched, is described below.

a) The decoy electronics will be switched on with the pressure switch getting activated at a depth of 5 m after launch.
b) The decoy settles down at a pre-set depth.
c) Each event of transmission of a burst of signal, by the torpedo, is called a ping and the duration of the signal is called ping-length. The logic that the decoy should use for its operation will depend on the torpedo's mode of operation. The decoy can function in two modes: the jammer mode and the transponder mode. The decoy will be in listening mode for 10 s to determine the torpedo's mode of operation. During this time, a minimum of three pings are expected if the torpedo is in active mode. If no transmission is received during this time, the decoy will switch into jamming mode in which it continues to transmit wideband white noise until a pre-set time (say 30 s) elapses after which the decoy will again listen for 10 s. The process repeats if there is no transmission from the torpedo. If a burst is detected, the decoy responds by transmitting a signal that simulates the echo from a real moving target.

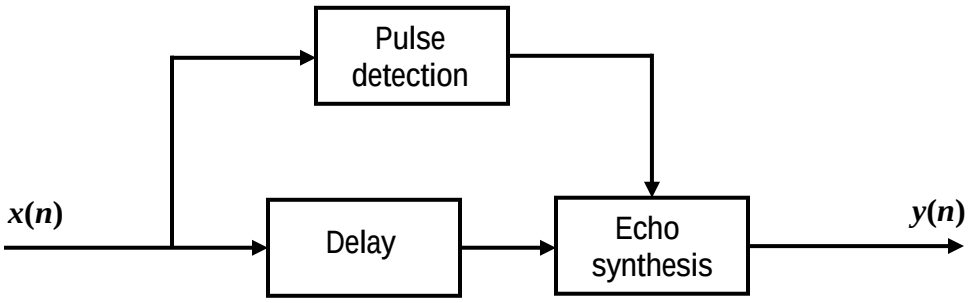


Fig. 2.1 Block diagram for decoy echo generation.

The decoy does not operate simultaneously in receiving and transmitting modes but operates either in a receiving or a transmitting mode at any instant of time. This is

because of the proximity in location of the receiving and the transmitting transducers and the difference in signal levels at the transducers for reception and transmission. Hence, the decoy operates in the receiving mode and processes the incoming signal sequence, $x(n)$, to detect the presence of a pulse, and subsequently processes the entire pulse after its detection to synthesize and transmit the echo signal, $y(n)$, from the moving target vessel, as shown in Fig. 2.1. During transmission of the simulated echo pulse, the decoy does not operate in the reception mode.

2.2 Decoy hardware

The basic hardware block diagram of the proposed system is shown in Fig. 2.2. A hydrophone acts as the receiving transducer and projector as the transmitting transducer. The signal-conditioning block includes the pre-amplifier and the band-pass filter. The digital signal processing block sends the digital data simulating the ship generated noise or the echo to the DAC. The output of the DAC is fed to a power amplifier before transmission.

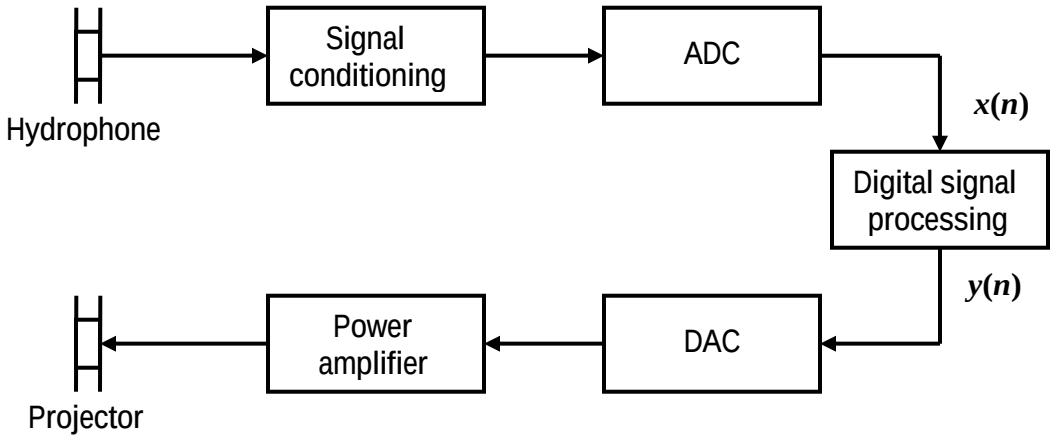


Fig. 2.2 Proposed block diagram of decoy electronics.

The functional block diagram of the digital signal processing block is shown in Fig. 2.3. During each 10 s of listening, the decoy operates continuously trying to detect an incoming signal pulse. If no pulse is detected during this period, the decoy acts in its

jammer mode in which case it sends the data from a noise generator to the DAC for next 30 s, otherwise it switches to its transponder mode in which a suitable echo is generated and sent to the DAC. The noise generator may consist of a digital memory that stores the recorded digital version of the actual ship-generated noise. The process of echo generation in the decoy involves basically the detection of the incoming signal and, subsequently processing the input signal, if detected, to synthesize the echo. A delay in the signal path compensates for the delay in pulse detection, and avoids the possibility of loss of a part of the received pulse during echo synthesis. In order to ensure a better probability of detection of this synthesized echo in the torpedo, the SNR of the input signal, detected by the decoy, is enhanced before the echo simulation. The echo simulation involves the following operations to be performed on the sequence of time-domain data:

1) the Doppler shift due to the movement of the ship, and
2) the pulse extension due to the size of a real target.

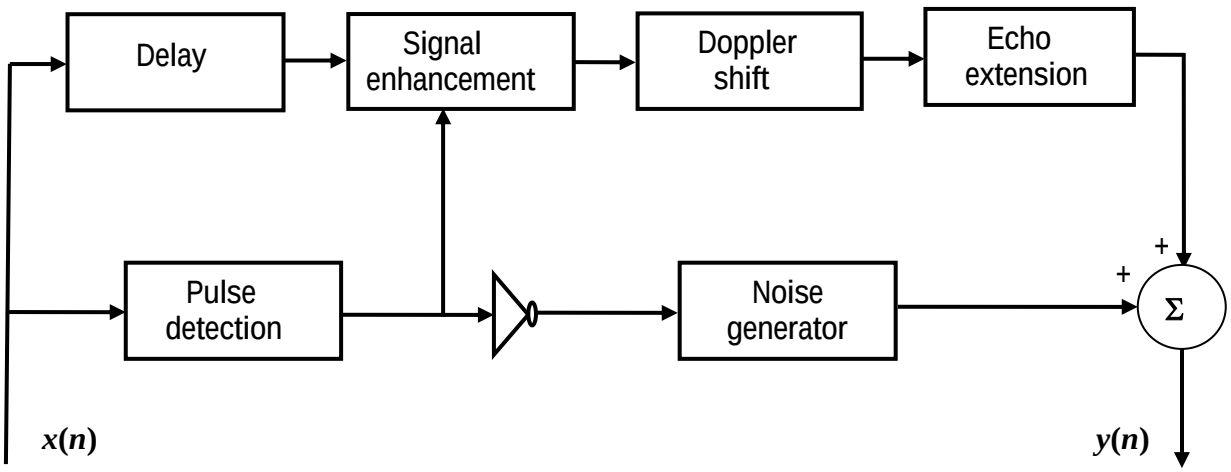


Fig. 2.3 The digital signal processing block of Fig. 2.2.

It is to be noted that driving the projector with the synthesized echo takes a lot of power. Hence, in order to conserve the battery power of the decoy, pulse detection should have a low probability of false detection.

2.3 Demodulation schemes employed in torpedo

The simulated echo transmitted by the decoy is processed in the torpedo to detect the presence of a target. The demodulation schemes, in recent torpedoes, are correlations and FFT processing depending on the types of signals that are used by the torpedo. Since different signals provide different resolutions in range and Doppler, the torpedo detection process employs different methods to resolve the ambiguity in finding the range and velocity of a target. The ambiguity in detection process for the case of a point target is described below with respect to PCW, LFM and CFS signals [4], [5].

The PCW signal has a poor performance in resolving range when a cross-correlator receiver is employed. This is because the envelope of the correlator output spreads over a duration of twice the pulse width of the transmitted signal. For example, with a PCW signal of pulse width 30 ms, the envelope of the cross-correlator output spreads for 60 ms. Moreover, the method is very sensitive to a shift in Doppler. That is, the peak of the correlator output falls by 3 dB with a shift of the signal frequency by just 30 Hz. This indicates that a large number of references are required for a correlator receiver to detect a target accurately with respect to its Doppler. However, adopting a FFT processing instead of correlation, the problem of resolving the Doppler shift is solved in case of PCW signal though the ambiguity in finding the range of the target still remains.

The LFM signal, on the other hand, offers a better range resolution when a cross-correlator receiver is employed to detect the signal. This is because of the pulse-compression characteristic, of the signal, in time at the output of the correlator. For example, a LFM signal of frequency sweep of 2000 Hz can resolve a range corresponding to time 0.5 ms (=1/2000 s). Moreover, the method is less sensitive to a shift in Doppler. That is, the peak of the correlator output falls by 3 dB with a shift of the center frequency of the signal sweep by over 500 Hz. This indicates that a few references are required for a correlator receiver to detect a target accurately in range. Thus, a correlation processing is adopted to determine the range of a target, in case of LFM signal whereas the Doppler ambiguity still remains.

The CFS signal has some of the merits of both the PCW and LFM signals. This signal provides a feature for an electronic counter-counter measure (ECCM) in the

torpedo. This signal is used when the torpedo reaches at a close distance from the target. A cross-correlation receiver is used in the torpedo for detection of such a signal.

Chapter 3

SIGNAL DETECTION AND ECHO SYNTHESIS

As mentioned in the previous chapter, the decoy generates and transmits an echo whenever it receives a signal from the torpedo during its active mode of operation. Thus, the process of echo synthesis is required to be carried out on the entire length of the detected signal. The different operations involved in the complete process of echo generation, as shown in Fig. 2.3, are i) pulse detection, ii) SNR enhancement, iii) Doppler shifting, and iv) echo extension. This chapter describes each individual operation of the echo generation process independently with a given sequence of input time domain data for that operation.

3.1 Pulse detection

The sequence of data samples, $x(n)$, is passed through a short-time energy detector and the output $\varepsilon(n)$ of the energy detector is given to a hysterisis comparator with an upper threshold and a lower threshold, to declare the presence of the incoming ping. Pulse detection is shown in Fig. 3.1. The short-time energy, $\varepsilon(n)$, is obtained by W-point moving average

$$\varepsilon(n) = \frac{1}{W} \sum_{l=0}^{W-1} x^2(n-l) \qquad n = 0, 1, 2, \ldots \tag{3.1}$$

The threshold is set on the basis of an estimate of background noise energy and an estimate of SNR. Generally, the SNR estimate is fixed and the noise energy estimate is to be made by the decoy. If the noise energy estimate is E_n, then the upper and lower threshold values are given by

$$\theta_1 = \alpha_1 E_n \tag{3.2}$$

and

$$\theta_2 = \alpha_2 E_n \tag{3.3}$$

respectively, where the constant α_1 is related to the minimum detectable SNR (in dB) ρ as

$$\rho = 10\log_{10}\alpha_1 \tag{3.4}$$

The factor α_2 is to be set to ensure that pulses are not missed, without having frequent false detection. To further reduce the possibility of false detection, the detected pulse width must exceed a set value. The block of the input sequence $x(n)$, corresponding to the detected pulse width, is sent to the echo synthesis block.

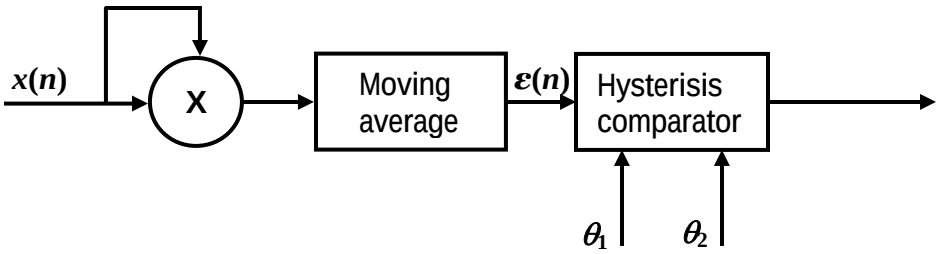


Fig. 3.1 Pulse detection block diagram.

The noise energy estimate, E_n, is either fed to the decoy during its launch or is calculated by the decoy itself after it is launched. The decoy, after it is launched, remains in the listening mode for 10 s and the mean-square value of the input samples during this entire time is considered as the estimated mean-square value of the noise. For a minimum distance of R between the torpedo and the decoy, the minimum pulse repetition interval (PRI) used by the torpedo sonar, is given as

$$\text{PRI} = \frac{2R}{c} \tag{3.5}$$

that is equal to 2.7 s if R = 2 km and c = velocity of acoustic signal in water = 1500 m/s. Thus, there can be a contribution of signal in this estimated noise value if the torpedo is in active mode. Assuming the active pulse width to be 100 ms (maximum), the minimum distance of the torpedo at the time of launching the decoy to be 2 km, and if the first pulse arrives at the decoy in the beginning of the observed time of 10 s, the decoy will receive signal for a maximum of four pulses, i.e., 400 ms in an entire duration of 10 s. Hence, the presence of pings can be considered as acceptable during the estimation of background noise.

The moving average length W for the energy detection is chosen as a compromise between the effect of non-stationary, spiky noise and the rise/fall time of the detector output in the presence of a signal pulse. Considering the minimum pulse width of the signal as 5 ms, the length W is chosen corresponding to a time duration of 2.5 ms.

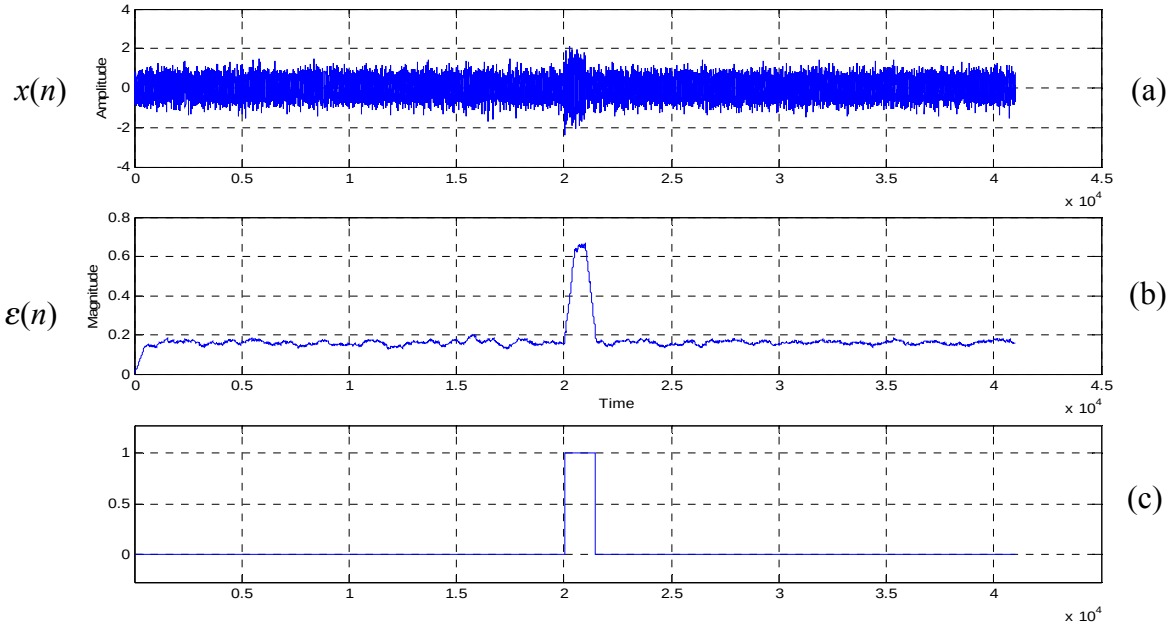


Fig. 3.2 Pulse detection: (a) received signal, (b) Short-time energy detector output, (c) Detected pulse (x-axis shows time index with f_s = 200 k sa/s).

After the rising edge of the signal pulse is detected, a minimum duration, of 2.5 ms, is observed to decide whether the pulse is present or absent. If a continuous break of 2.5 ms occurs, the pulse is considered no longer present otherwise the pulse width extends from the first rising edge to the last threshold crossing (i.e., the falling edge) of the energy detector output. The detection of a new pulse starts if the pulse width corresponds to less than 2.5 ms otherwise a sequence of data samples corresponding to the estimated time duration is sent for subsequent operation and the operation of pulse detection is resumed afresh.

It is to be noted that loss of any part of the incoming ping as the selected pulse may make the echo synthesis ineffective. On the other hand, including a non-ping part of input as part of the selected pulse only results in additional power drain during the

transmission of the synthesized echo. Hence, emphasis should be on not missing any part of the incoming ping.

Let the rising edge of the pulse, as detected, be denoted at a time index p, but then this edge of the pulse may be detected at a later point than the actual point of occurrence of the pulse due to the presence of noise and the rise time delay of the energy detector. Hence, the beginning of the pulse is considered 2.5 ms previous to the detected index, p. Similarly, a time duration of 2.5 ms after the detected end of the pulse is included to constitute the actual pulse. Thus, the obtained pulse width can range from 10 ms to 105 ms corresponding to the actual pulse width ranging from 5 ms to 100 ms.

Figure 3.2(a) shows the simulated noisy signal received by the decoy, and the detected pulse is shown in Fig. 3.2(c). A CW signal was used with a pulse width of 5 ms with an SNR of 5 dB.

3.2 SNR improvement

The decoy processes the received signal to reduce the additive noise accompanied with the signal while traveling in the channel so that the transmitted signal is corrupted by only one-way additive noise when received by the torpedo thus improving the probability of detection of the signal in the torpedo. This section describes the spectral noise subtraction method for enhancing the signal corrupted by a broadband noise [9], [10]. The assumption made in this method is that the power spectrum of a signal corrupted by uncorrelated noise is equal to the sum of the signal spectrum and the noise spectrum.

The method essentially consists in computing the power spectrum of the segment of the signal, $P_{s+n}(\omega)$, and subtracting from it an estimate of the noise power spectrum, $P_n(\omega)$. The time domain signal sequence is passed through an FFT block and the power spectrum, $P_{s+n}(k)$, is obtained by squaring each component of the resulting magnitude spectrum. The phase, $\theta(k)$, of the DFT of this input signal is retained for re-synthesis. The estimate of the noise spectrum, $P_n(k)$, is obtained by averaging the noise power spectra from several frames of input data during a period when no signal is assumed to be present.

$$D(k) = P_{s+n}(k) - \alpha P_n(k) \tag{3.6}$$

$$Q(k) = D(k), \qquad \text{if } D(k) > \beta P_n(k) \tag{3.7}$$

$\beta P_n(k)$, otherwise

with $\alpha \geq 1$, and $0 < \beta << 1$, where α is the subtraction factor, β is the spectral floor parameter and $Q(k)$ is the modified signal power spectrum as shown in Fig. 3.3. The complex spectrum of the enhanced signal is obtained as

$$X(k) = \sqrt{Q(k)}\, e^{j\theta(k)} \tag{3.8}$$

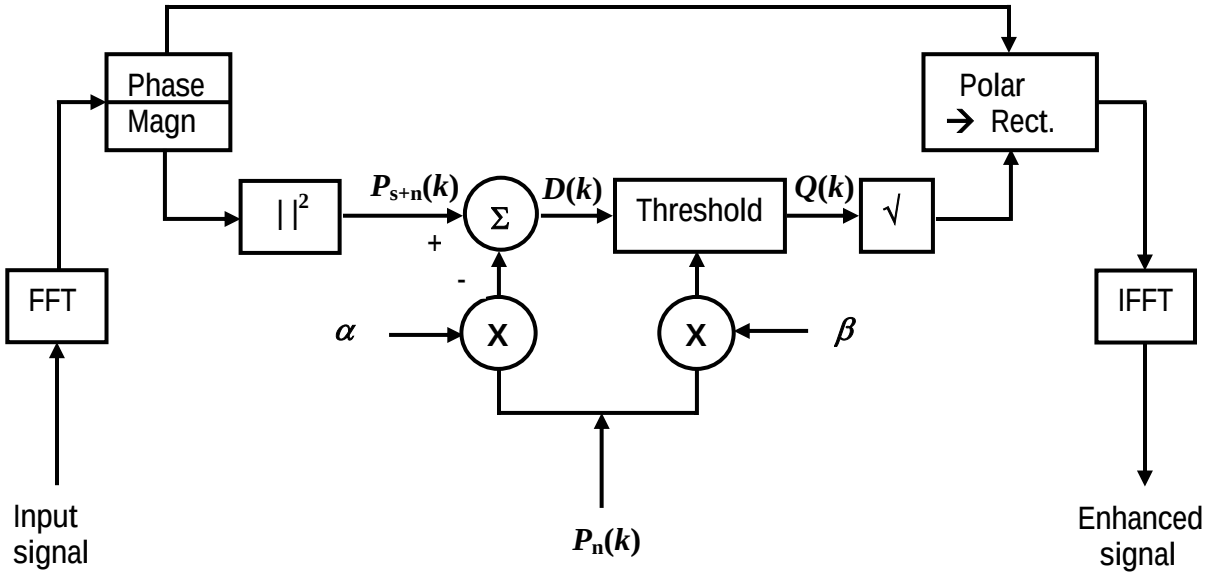


Fig. 3.3 Spectral noise subtraction method [10].

For $\alpha > 1$, the subtraction removes all of the broadband noise by eliminating most of the wide peaks. However, this causes some of the components of $Q(k)$ to become negative. With $\beta > 0$, the spectral components of $Q(k)$ are prevented from descending below the lower bound $\beta P_n(k)$. For $\beta << 1$, the added broad band noise level is also much lower. It has been shown in [10] that the values of α should change linearly from 6 to 1 with the SNR ranging from -5 to 20 dB. However, in the present case, $\alpha = 2$ has been chosen as the reasonable value considering that seldom the SNR goes below 5 dB. Furthermore, $\beta = 0.001$ ensures a 30 dB attenuation of the broadband noise.

Figure 3.4 shows a signal in white Gaussian noise as well as the enhanced signal that is obtained after implementing the spectral noise subtraction method on the noisy

signal. The signal used was a CW signal of frequency 40 kHz and amplitude unity with an SNR of 5 dB.

3.3 Doppler shifting

In the process of Doppler shifting, the decoy introduces, in the received signal, a certain frequency shift that should have taken place had the signal got reflected after hitting a moving target. The Doppler shift caused by the motion of the torpedo does not have to be considered because it will take place even for the synthesized echo. Assuming that the target ship is moving with a radial velocity v, the incoming spectral component f should get shifted to

$$f' = f(1- \frac{2}{c} v) \tag{3.9}$$

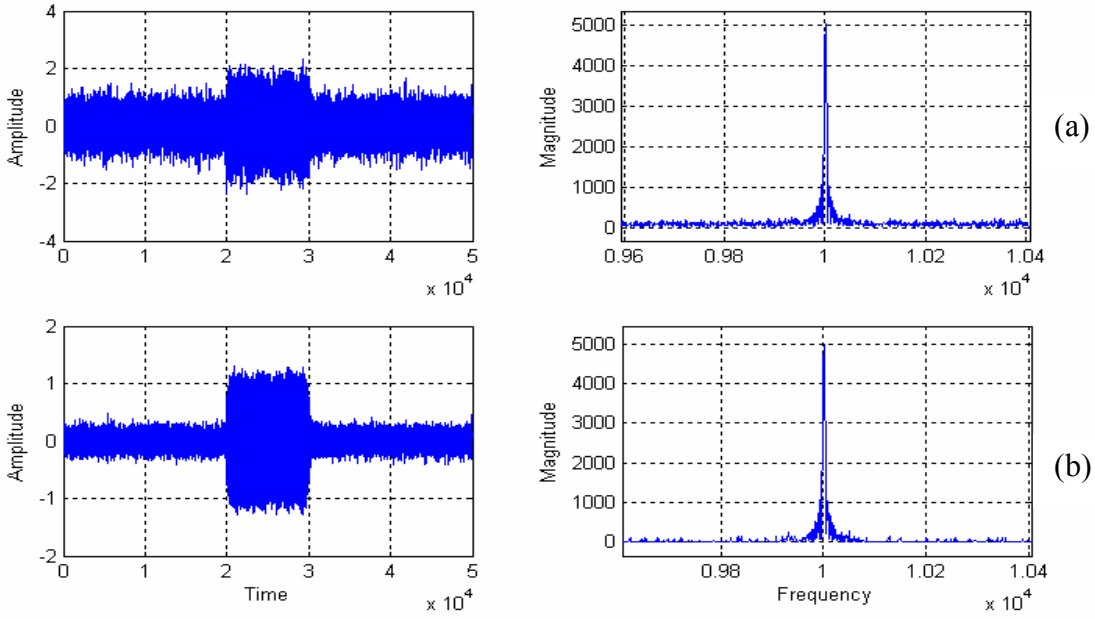


Fig 3.4 Signal enhancement using spectral noise subtraction: (a) Noisy signal and its magnitude spectrum, and (b) Enhanced signal and its magnitude spectrum (α = 2 and β = 0.001). Time and frequency axes show the indices corresponding to f_s = 200 k sa/s.

The torpedo declares the target if v has a non-zero positive value. If v is zero, the reflection is considered to be from a stationary object such as a rock whereas a negative v indicates that the target is approaching which is very unlikely in reality. The factor by

which a particular frequency component of the received signal has to be scaled is given as

$$\alpha = (1 - \frac{2}{c} v) \tag{3.10}$$

and the pulse width T becomes

$$T' = \frac{T}{\alpha} \tag{3.11}$$

Thus, during the simulation of the echo in the decoy, the Doppler shift in the received signal can be introduced in two ways: (a) in frequency domain by scaling the frequency of the spectral components by a factor or conversely, (b) in time domain by scaling the duration of the signal by the inverse of the same factor.

From Eqn. 3.10, it is seen that the factor α, decreases with an increase in the velocity of the target vessel v. The approaches to introduce the Doppler shift in the signal received by the decoy, in time domain and in the frequency domain, are further discussed below.

3.3.1 Frequency domain processing

In this approach, DFT is computed on the time domain input sequence of length N, and the spectral samples are shifted in position according to the amount of Doppler shift needed. Since the signal transmitted by the torpedo is assumed to have the frequency components between $f_{min} \approx 15$ kHz and $f_{max} \approx 85$ kHz, the corresponding frequency components of the signal received by the decoy will range between

$$f_l = f_{min}(1+ \frac{2}{c} u_{min}) \tag{3.12}$$

and

$$f_h = f_{max}(1+ \frac{2}{c} u_{max}) \tag{3.13}$$

respectively, where u_{min} (u_{max}) is the minimum (maximum) velocity of the torpedo. Considering $u_{min} = 0$ and $u_{max} = 25$ m/s (≈ 50 knots), the DFT outputs corresponding to this frequency band, i.e., $f_l = f_{min}$ and $f_h = 1.03 f_{max}$, are shifted to incorporate the required Doppler shift. Frequency-shifting operation is done for the first $N/2+1$ samples of the DFT output and complex conjugate symmetry of DFT is used for obtaining the rest of the samples.

The frequency index, corresponding to the frequency f is given as

$$k = \frac{fN}{f_s} \tag{3.14}$$

and the corresponding frequency index, after Doppler shifting, is

$$k' = k\alpha \tag{3.15}$$

where α is related to the ship velocity v by Eqn. 3.10.

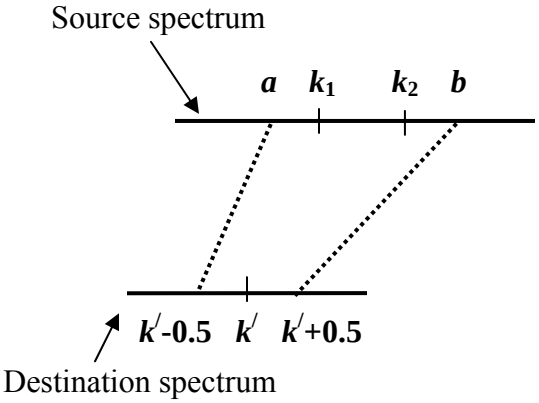


Fig. 3.5 Frequency shifting by frequency segment mapping.

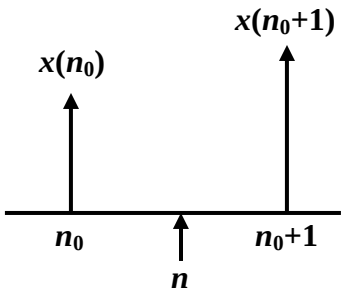


Fig. 3.6 Method of linear interpolation.

Let k_l and k_h denote the frequency indices corresponding to the two edges of the signal band, then the operation of Doppler shifting was achieved by shifting in position these frequency bins to the frequency bins between k_l' and k_h', corresponding to the new band. Two schemes, called (i) frequency sample mapping, and (ii) frequency segment mapping schemes, were investigated to compute the frequency components in the new band.

Let us call the received signal as the source and the synthesized echo as the image. In frequency sample mapping method, each spectral sample in the image is obtained from the source as

$$X'(k') = X(k) \tag{3.16}$$

where

$$k = \text{round}(k'/\alpha) \tag{3.17}$$

In this mapping scheme, not all the spectral samples necessarily contribute to the output. In case of spectrum being highly impulsive, the output spectrum may not represent the input properly. The total power of the two spectra may differ.

In the frequency segment mapping method, each spectral sample in the image spectrum is obtained as numerical integral of the spectral density over the corresponding frequency interval in the source spectrum. Suppose, we need to find $X'(k')$, then we compute, as shown in Fig. 3.5,

$$a = (k'-0.5)/\alpha, \tag{3.18}$$

$$b = (k'+0.5)/\alpha \tag{3.19}$$

and finally,

$$X'(k') = X(k_1)[\, k_1 - a + \frac{1}{2}\,] + \sum_{k=k_1+1}^{k_2-1} X(k) + X(k_2)[b - k_2 + \frac{1}{2}\,] \tag{3.20}$$

where k_1 = round(a) and k_2 = round(b). The central summation term will contribute only if $k_2 \geq k_1+2$. Although computationally more involved, this technique is free from the problems associated with sample mapping.

3.3.2 Time domain processing

In this approach, the Doppler shift in a signal sequence is introduced in time domain by scaling the length of the sequence. Given the length of the input signal sequence as L, the length of the new Doppler shifted sequence, from Eqn. 3.11, will be

$$L' = \frac{L}{\alpha} \tag{3.21}$$

We have attempted this scaling in time domain by 2-point interpolation. Let n' be the sample index in the image, and it corresponds to a sample index

$$n = n'\alpha \tag{3.22}$$

in the source, as shown in Fig. 3.6. The image sample is then obtained as the weighted average of two source samples as

$$x'(n') = (1 - \beta)x(n_0) + \beta\, x(n_0 + 1) \tag{3.23}$$

where n_0 = int(n) and $\beta = n - n_0$. The main justification for restricting the interpolation to 2-point is its computational simplicity.

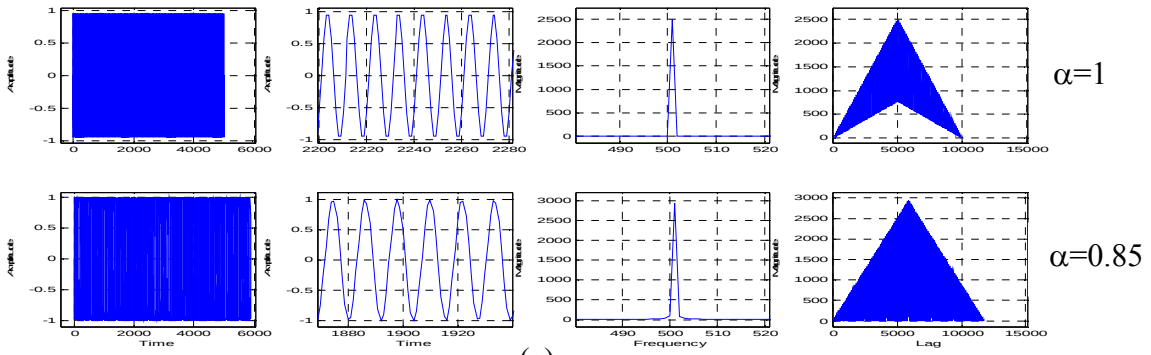
3.4 Test results with Doppler shifting

The performance of the two approaches of implementing Doppler shift has been investigated for PCW, LFM, and CFS signals having duration of 2.5-100 ms with a 10-90 kHz range. Although the frequency components of the signal, in reality, could be down-shifted by a minimum value of $\alpha = 0.98$ corresponding to a maximum speed of the target vessel as 15 m/s (≈ 30 knots), the investigation has been carried out for α over a range of 0.85-0.98. The resulting time domain sequences and their spectra, for different signal durations, frequencies, and ranges of shift factor α, were graphically compared. Also, the zoomed versions of an arbitrary section of the time domain sequence, for $\alpha = 1$, and others corresponding to different values of α, are shown to indicate how closely the Doppler shifted signal, generated by time domain approach and frequency domain approach, for each value of α, resembles the original signal. Furthermore, since the most common detection scheme used in the torpedo is the cross-correlation of the echo with the stored references, the results of cross-correlation of the sequences obtained by the Doppler processing with those Doppler shifted references were also observed. The Doppler shifted references are generated by using the same input signal, used for Doppler processing, but with modified frequency and duration that will actually result in the input signal after the Doppler processing. For example, if a PCW signal, having frequency f and time duration T, is used for Doppler processing, the output of the Doppler processing should result to another PCW signal with frequency f' and duration T'. Thus, the reference signal is generated using a PCW signal having frequency f' and duration T'.

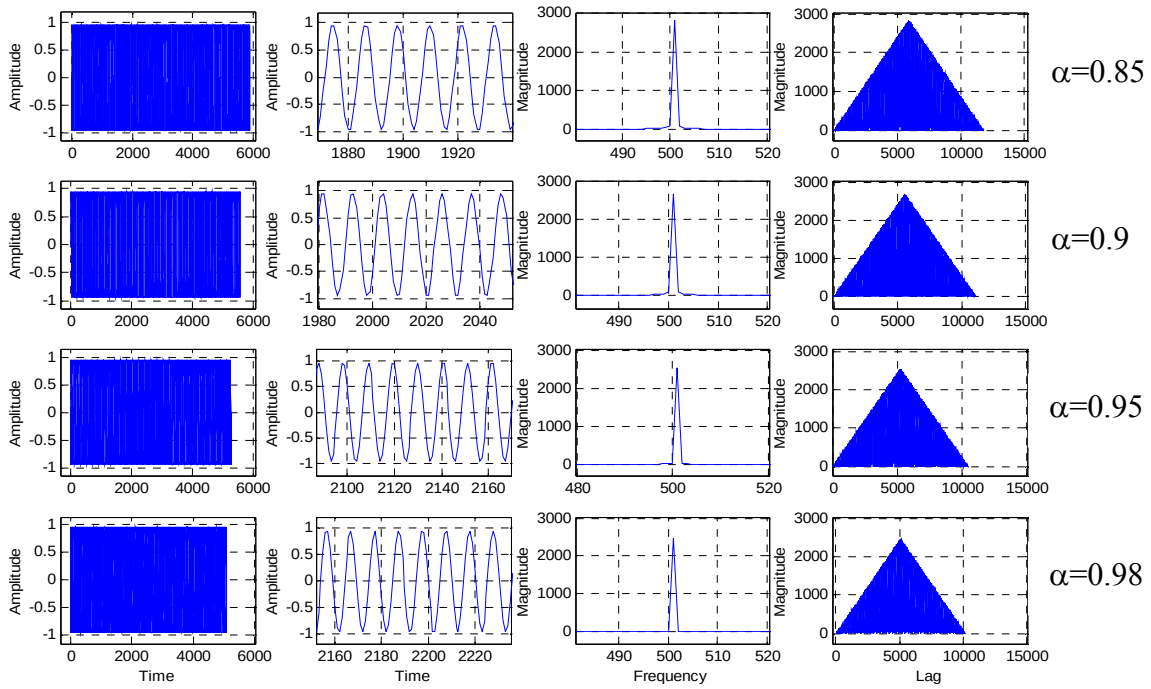
3.4.1 Doppler shifting using time domain processing

Figures 3.7-3.9 show the results of introducing different Doppler shifts to three different types of signal sequences using time domain approach. The time domain sequences, a zoomed version of a section of each, the magnitude spectra, and the auto-correlation output for the original PCW signal having a frequency of 20 kHz and duration $L = 5000$, and a single Doppler shifted reference for $\alpha = 0.85$ are shown in Fig. 3.7(a). Figure 3.7(b) shows the resulting time domain sequences, obtained by the time domain approach, their zoomed versions, spectra, and their cross-correlation results with the Doppler shifted

references corresponding to four different values of α. Similarly, Figs. 3.8 and 3.9 show the results for the LFM and CFS signals, respectively. The original LFM signal has start frequency 40 kHz, sweep 2 kHz, and L = 10000 while the original CFS signal has seven discrete frequencies starting from 60 kHz, and L = 20000.



(a)



(b)

Entire signal Zoomed section Magnitude spectrum Cross-corr. output

Fig. 3.7 Doppler shifting using time domain approach: (a) Original PCW signal with L = 5000 and f = 20 kHz, and its Doppler shifted reference with α = 0.85, and (b) results of Doppler shifting on the original signal with different values of α. Time and frequency axes show the indices corresponding to f_s = 200 k sa/s.

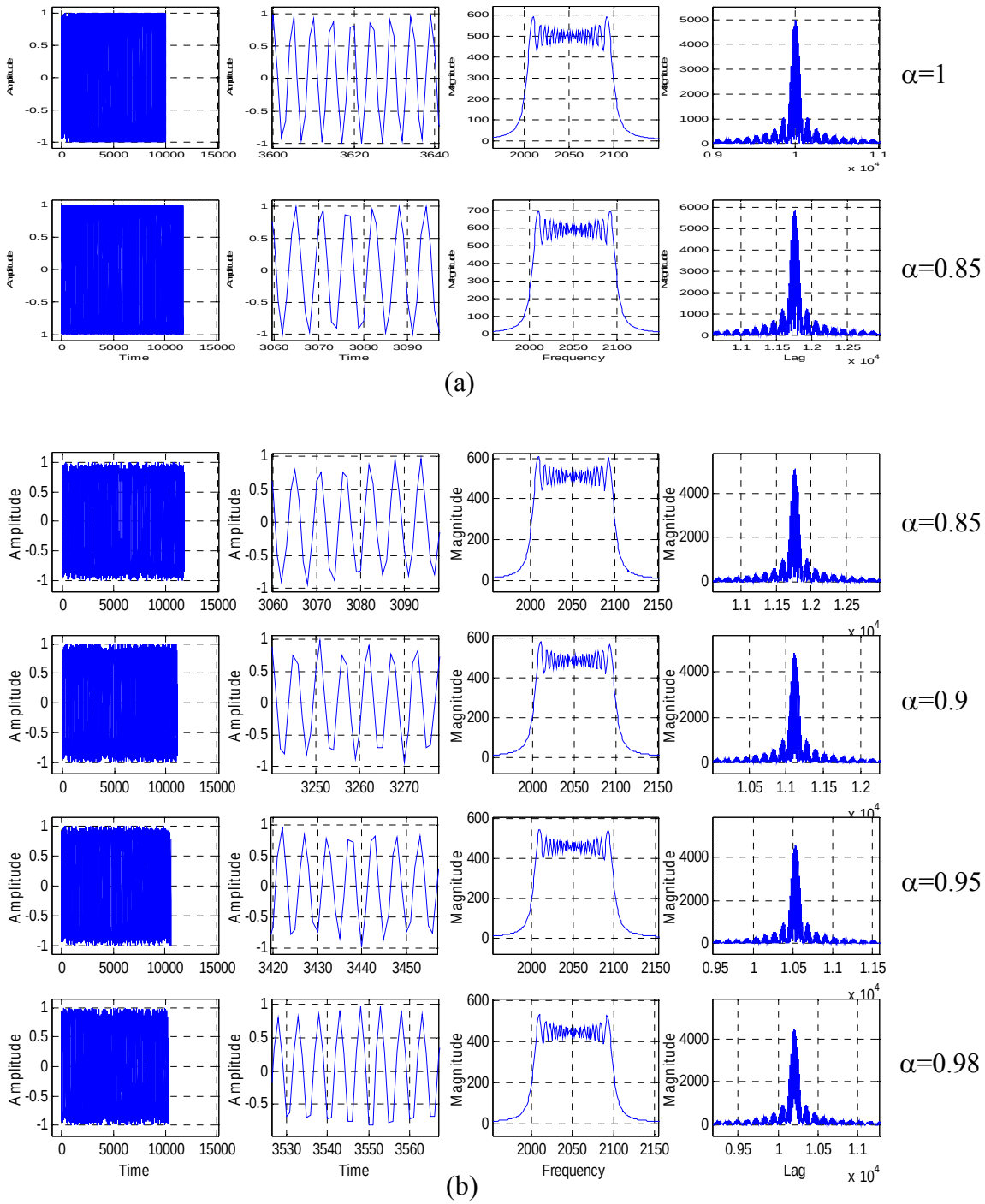


Fig. 3.8 Doppler shifting using time domain approach: (a) Original LFM signal with L = 10000 and f_l = 40 kHz, sweep 2 kHz, and its Doppler shifted reference with α = 0.85, and (b) results of Doppler shifting on the original signal with different values of α. Time and frequency axes show the indices corresponding to f_s = 200 k sa/s.

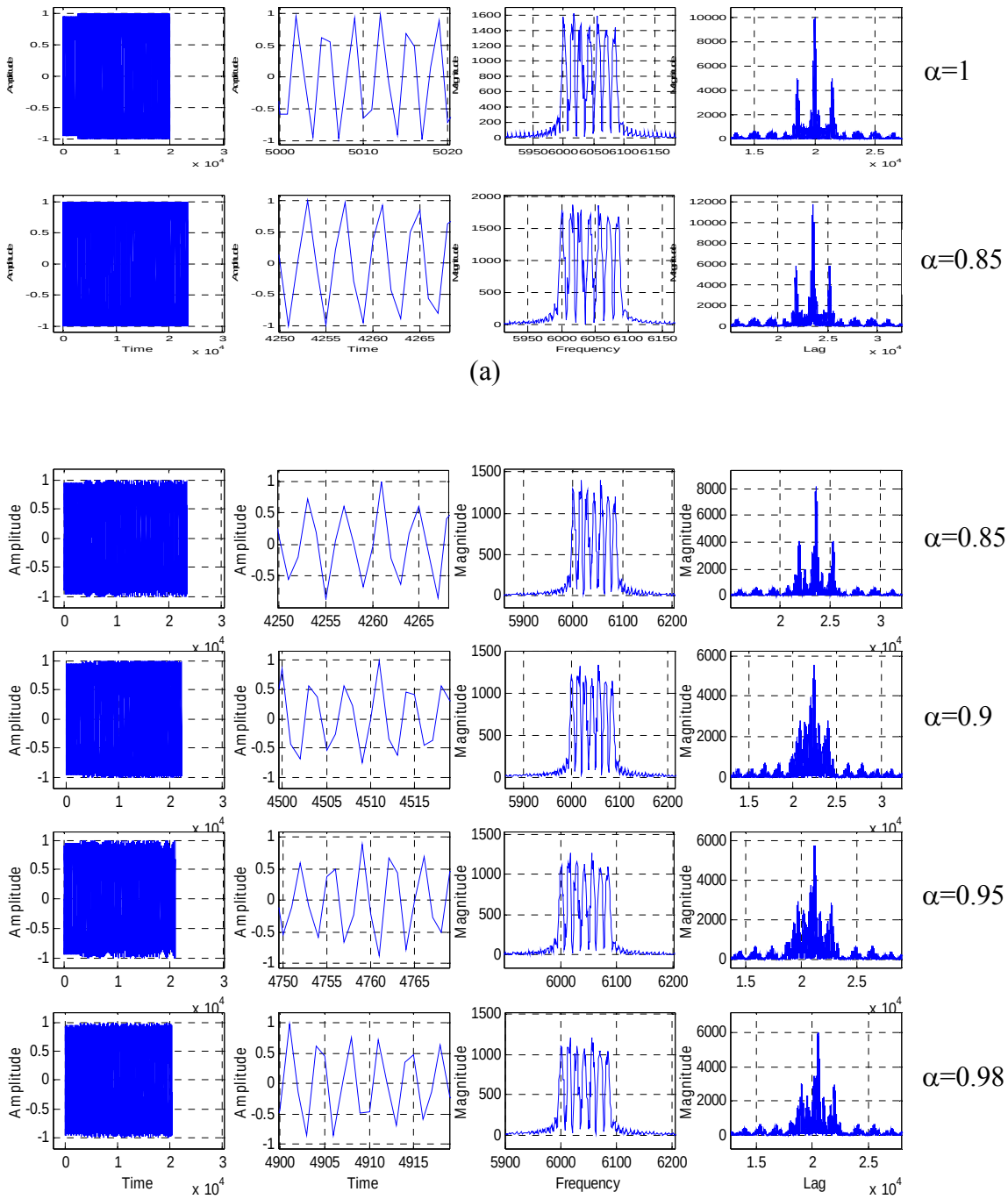
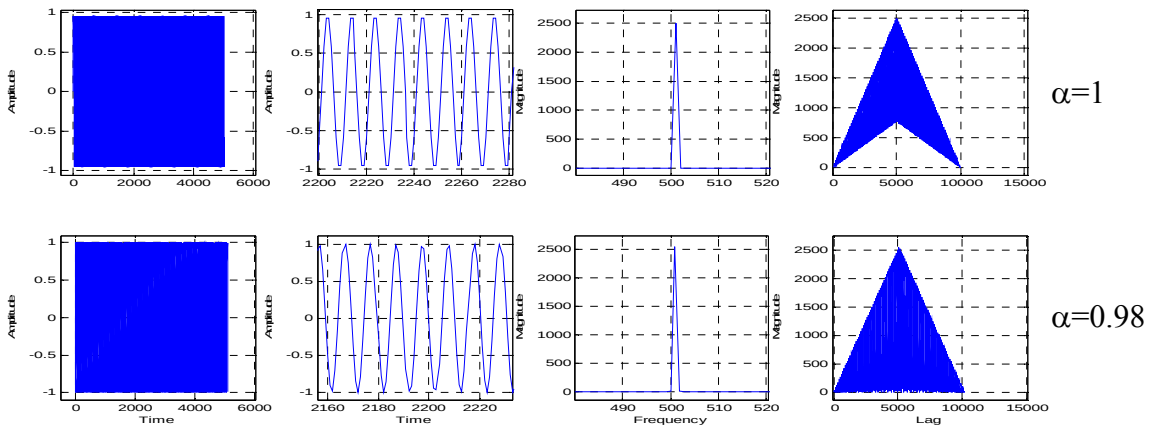
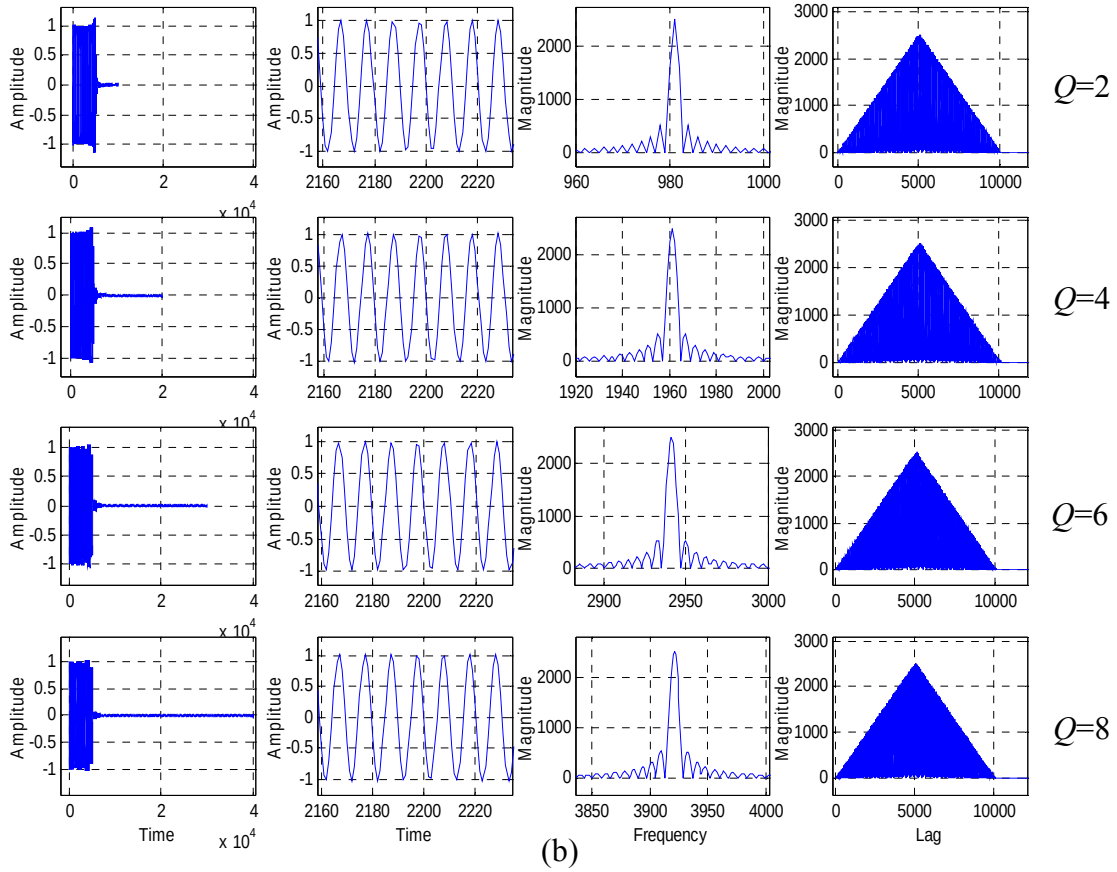


Fig. 3.9 Doppler shifting using time domain approach: (a) Original CFS signal with L = 20000 and f_0 = 60 kHz, seven discrete frequencies, and its Doppler shifted reference with α = 0.85, and (b) results of Doppler shifting on the original signal with different values of α. Time and frequency axes show the indices corresponding to f_s = 200 k sa/s.



(a)



(b)

Entire signal Zoomed section Magnitude spectrum Cross-corr. output

Fig. 3.10 Doppler shifting using frequency domain approach (sample mapping method): (a) Original PCW signal with L = 5000 and f = 20 kHz, and its Doppler shifted reference with α = 0.98, and (b) results of Doppler shifting on the original signal with different values of Q. Time and frequency axes show the indices corresponding to f_s = 200 k sa/s.

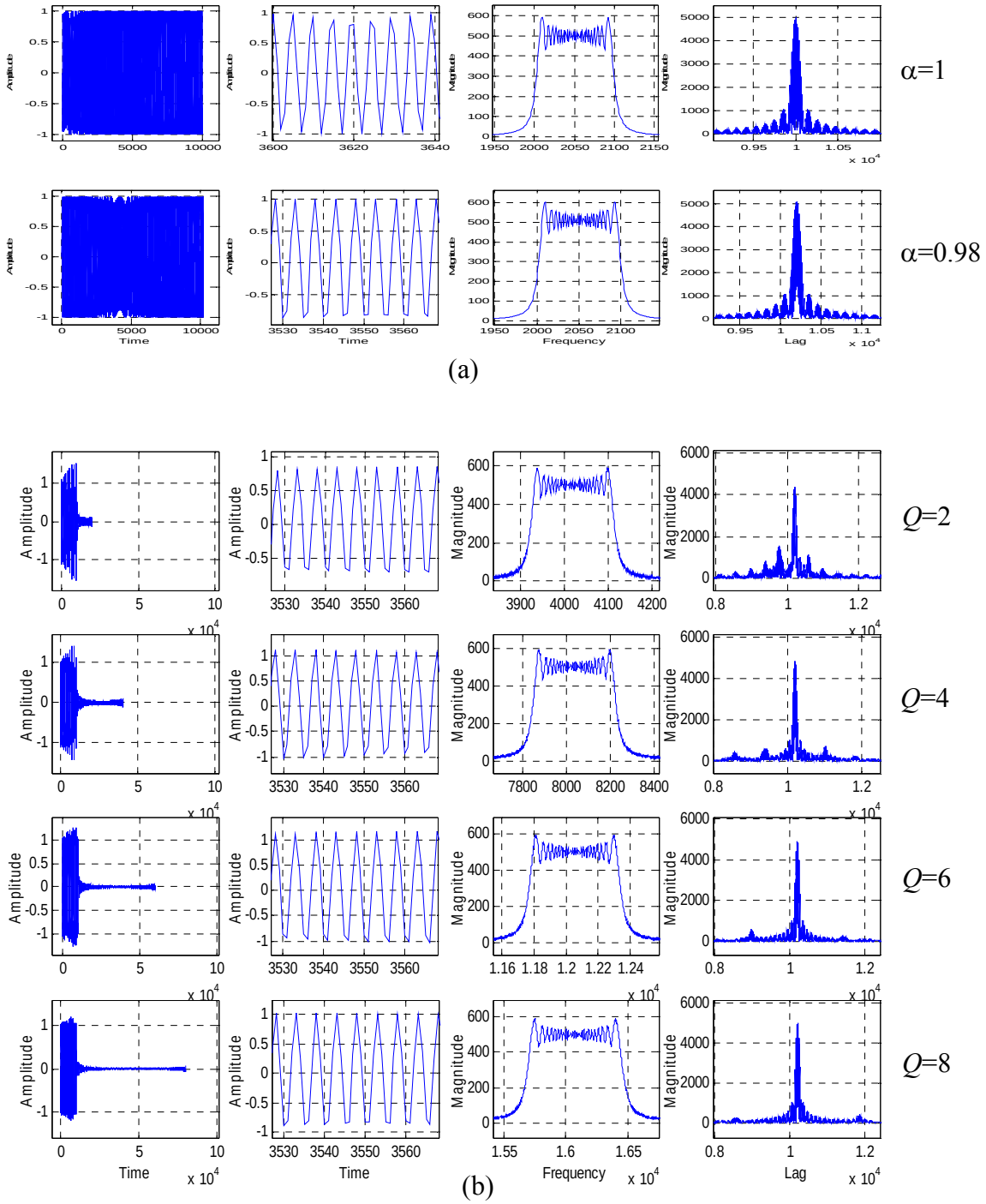


Fig. 3.11 Doppler shifting using frequency domain approach (sample mapping method): (a) Original LFM signal with L = 10000 and f_l = 40 kHz, sweep 2 kHz, and its Doppler shifted reference with $\alpha = 0.98$, and (b) results of Doppler shifting on the original signal with different values of Q. Time and frequency axes show the indices corresponding to f_s = 200 k sa/s.

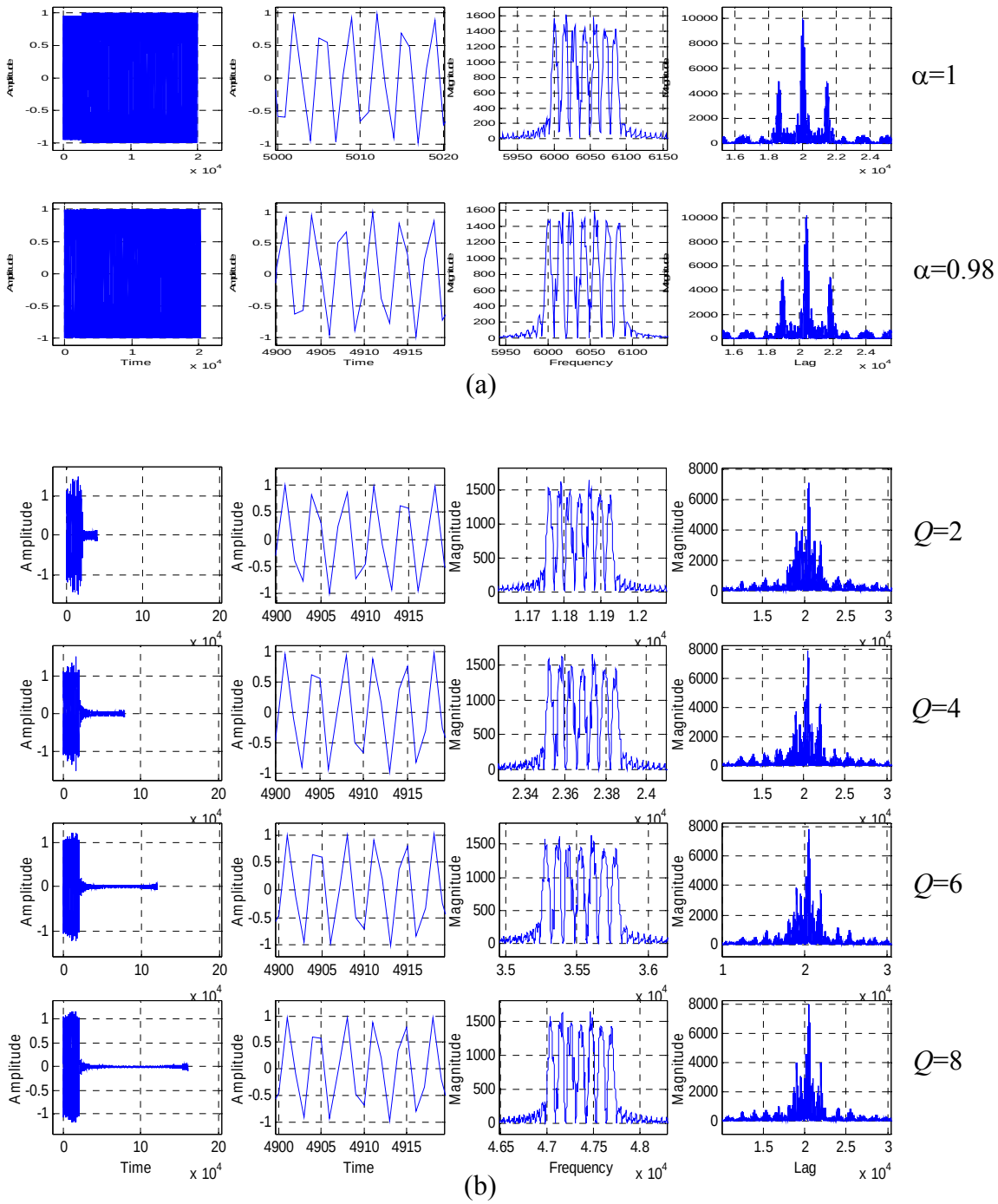


Fig. 3.12 Doppler shifting using frequency domain approach (sample mapping method): (a) Original CFS signal with L = 20000 and f_0 = 60 kHz, seven discrete frequencies, and its Doppler shifted reference with α = 0.98, and (b) results of Doppler shifting on the original signal with different values of Q. Time and frequency axes show the indices corresponding to f_s = 200 k sa/s.

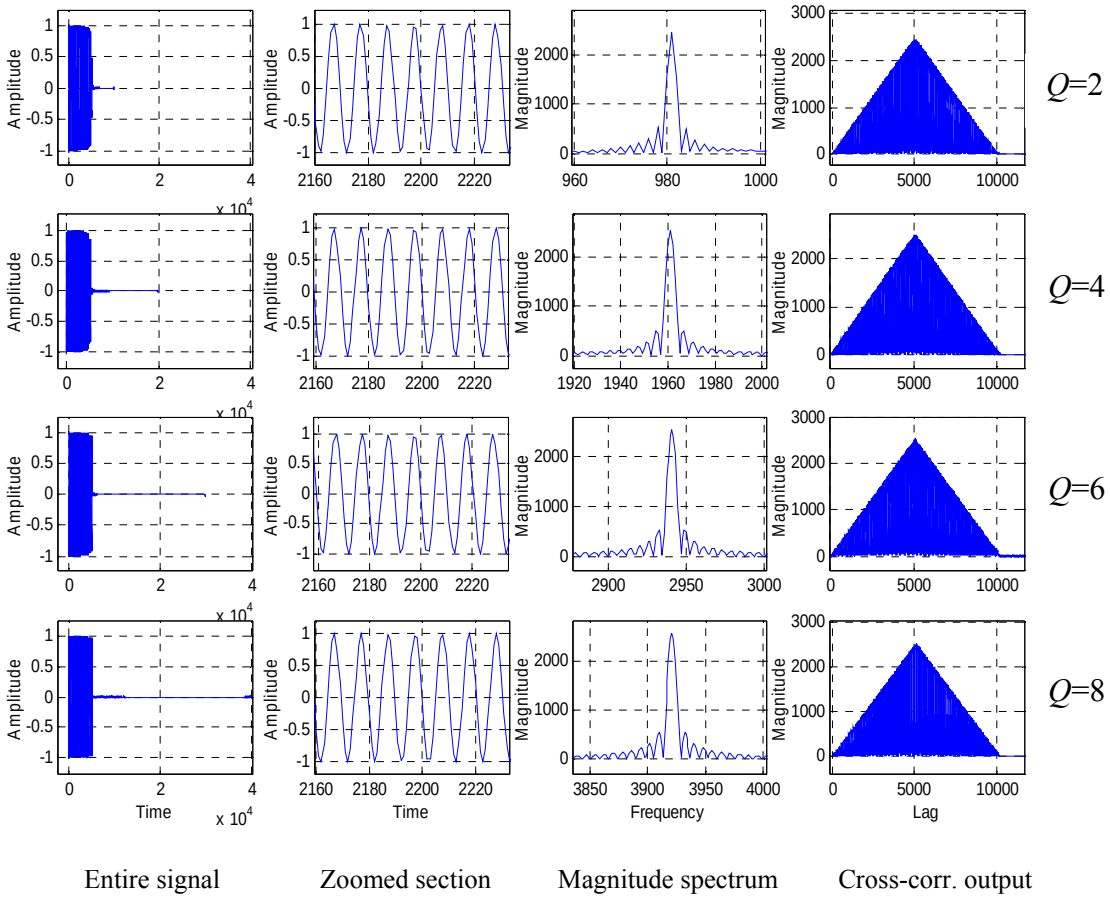


Fig. 3.13 Doppler shifting using frequency domain approach (segment mapping method): (a) Original PCW signal with L = 5000 and f = 20 kHz, and its Doppler shifted reference with α = 0.98, and (b) results of Doppler shifting on the original signal with different values of Q. Time and frequency axes show the indices corresponding to f_s = 200 k sa/s.

It is seen that imparting Doppler shift to any signal in time domain, with any given value of α, introduces not much distortion in the resulting time domain signal, except for a small reduction in the amplitude of the signal.

3.4.2 Doppler shifting using frequency domain processing

Doppler shift, in frequency domain processing, by down-shifting the spectrum of the time domain signal sequence by a factor α amounts to an increase in length of the resulting

time domain sequence by the factor, $1/\alpha$. Hence, the original time domain sequence should be zero-padded to a length equal to $1/\alpha$ times its original length or longer.

Investigations were carried out to see the effect of different lengths of zero padding, on wave distortion, for different values of α, using the frequency domain approach.

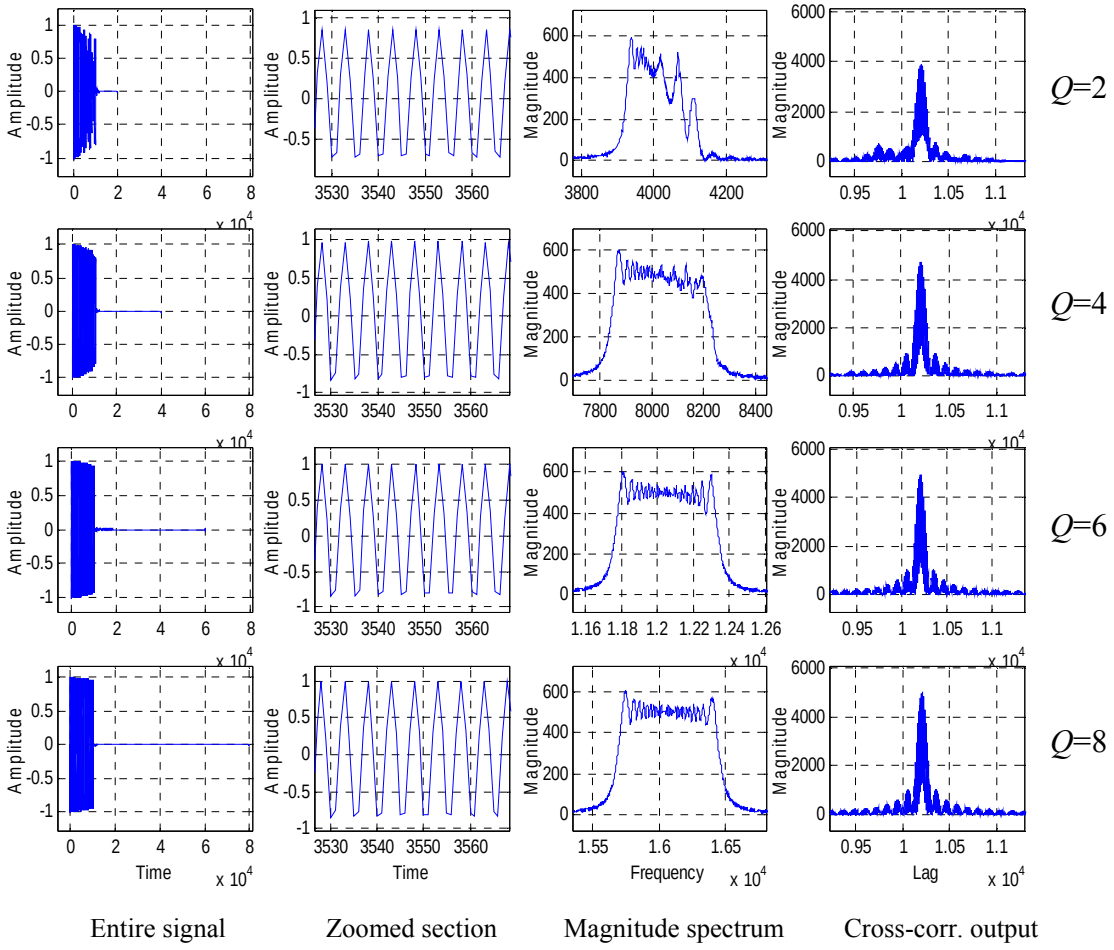


Fig. 3.14 Doppler shifting using frequency domain approach (segment mapping method): (a) Original LFM signal with L = 10000 and f_l = 40 kHz, sweep 2 kHz, and its Doppler shifted reference with α = 0.98, and (b) results of Doppler shifting on the original signal with different values of Q. Time and frequency axes show the indices corresponding to f_s = 200 k sa/s.

Figures 3.10-3.12 show the simulation results indicating the effects of employing different lengths of zero-padding to a time domain sequence when Doppler shifts are

implemented using frequency sample mapping. Sequence length L is padded to a length N, with $Q = N/L$. The cases of Doppler shifts, corresponding to $\alpha = 0.98$, are presented for sequences of three different types of signals with different frequencies and pulse widths.

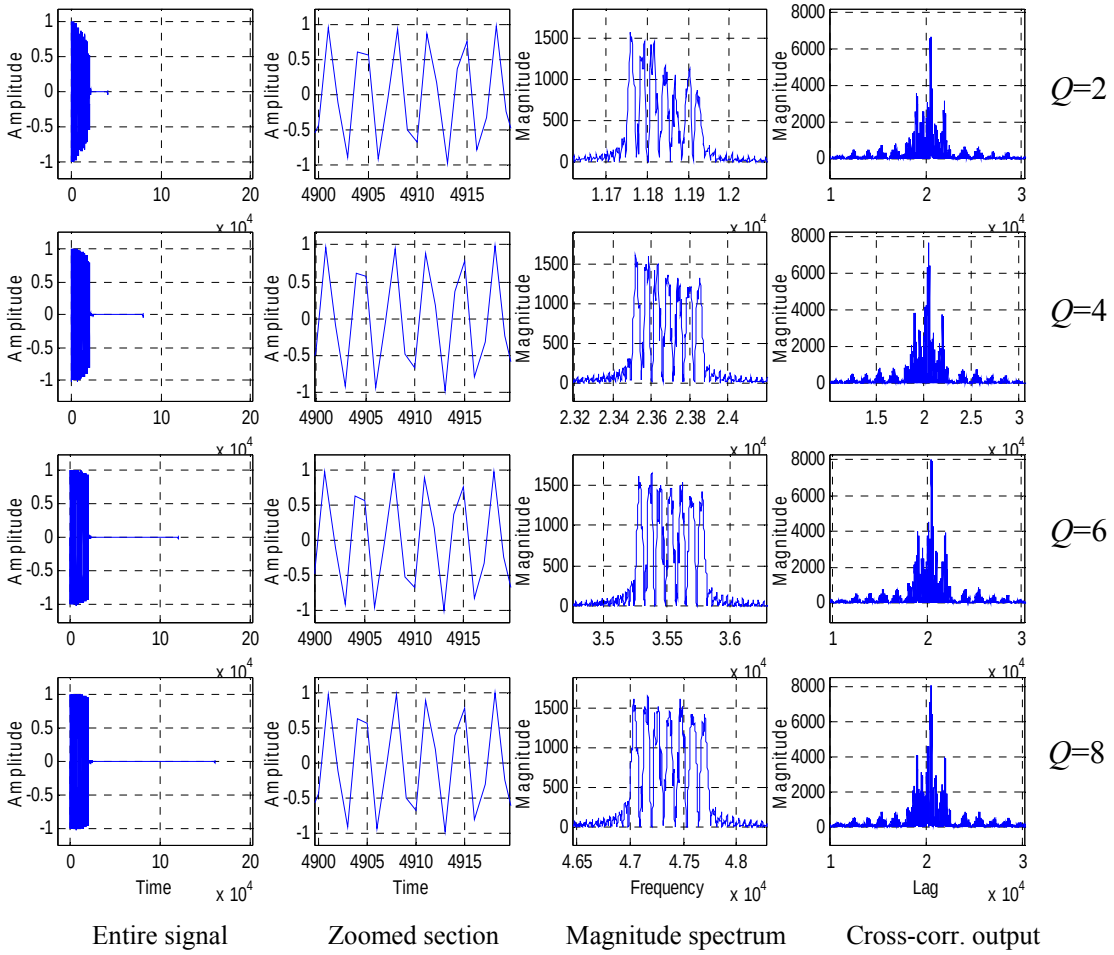


Fig. 3.15 Doppler shifting using frequency domain approach (segment mapping method): (a) Original CFS signal with $L = 20000$ and $f_0 = 60$ kHz, seven discrete frequencies, and its Doppler shifted reference with $\alpha = 0.98$, and (b) results of Doppler shifting on the original signal with different values of Q. Time and frequency axes show the indices corresponding to $f_s =$ 200 k sa/s.

Figure 3.10 presents a case when the frequency shift is imparted on a PCW signal of length, $L = 5000$. The time domain sequences along with zoomed versions of a small

section of each, the magnitude spectra, and the auto-correlation results of the original PCW signal and the Doppler shifted reference for $\alpha = 0.98$ are shown in Fig. 3.10(a). It can be seen that the resulting time domain sequence grows to a length L' (= 5102) that is $1/\alpha$ times L. Hence, the value of Q should strictly not be less than $1/\alpha$ that is 1.02 in this case. However, as shown in the Fig. 3.10(b), the resulting time domain sequence shows a significant reduction in overall amplitude and distortion towards the end even for $Q = 2$. The reduction in amplitude results from the fact that the total energy of the image spectrum is now less than that of the source spectrum due to the reduction in the width of the image spectrum without a proportional increase in its amplitude. The distortion in the resulting time domain sequence can be accounted for due to the poor spectral resolution of the source spectrum because of which the mapping of the frequency samples results in some abrupt changes in the image spectrum thereby resulting in an extended duration of the time domain signal. This distortion is observed in terms of the fluctuations in amplitude in the resulting sequence and increasing the coherence time. The distortion gradually decreases as Q increases and a reasonably good result is obtained by taking $Q = 8$ that requires a zero-padding of length 7 times the original length of the time domain signal. However, the overall amplitude of the resulting time domain sequence remains less.

Similarly, the simulation results, for LFM and CFS signals, are shown in Figs. 3.11 and 3.12, respectively. Reasonable result is obtained by taking $Q = 8$ for $\alpha = 0.98$, in the case of both LFM and CFS signals. The minimum acceptable values of parameter Q, obtained through simulation for different signals and values of α, are given in Table 3.1. The simulation results of applying segment mapping method of implementing Doppler shift, using signals and value of α similar to those employed for frequency sample mapping method, are shown in Figs 3.13-3.15.

Figure 3.13 presents a case when a frequency shift of $\alpha = 0.98$ is imparted on a PCW signal of length, $L = 5000$. As mentioned before, the value of Q should strictly not be less than $1/\alpha$ that is 1.02 in this case. However, as shown in the Fig. 3.13, the resulting time domain sequence shows a significant distortion towards the end even for $Q = 2$. The distortion in the resulting time domain sequence can be accounted for due to the poor spectral resolution of the source spectrum as well as the mapping of the segments of

progressively increasing lengths with frequency from the source spectrum. The mapping of the frequency segments by summing the samples from the source spectrum results in a somewhat progressive smoothening in the envelope of the image spectrum with increase in frequency. This distortion is observed in terms of the tapering in amplitude in the resulting sequence. The distortion gradually decreases as Q increases and a reasonably good result is obtained by taking $Q = 6$ that requires a zero-padding of length 5 times the original length of the time domain signal. Since each spectral sample in the image spectrum is obtained as the summation of the spectrum over a corresponding frequency interval in the source spectrum, the total energies in the source and the image spectra remain same and hence the amplitude of the resulting time domain sequence remains same as that in the original sequence.

Table 3.1 Values of $Q = N/L$ for low distortion in "frequency sample mapping" method of Doppler shifting with frequency domain processing.

Modulation type	Q			
	$\alpha = 0.85$	$\alpha = 0.9$	$\alpha = 0.95$	$\alpha = 0.98$
PCW	8	8	7	7
LFM	9	9	7	7
CFS	9	9	8	7

Table 3.2 Values of $Q = N/L$ for low distortion in "segment mapping method" of Doppler shifting with frequency domain processing.

Modulation type	Q			
	$\alpha = 0.85$	$\alpha = 0.9$	$\alpha = 0.95$	$\alpha = 0.98$
PCW	6	6	5	5
LFM	7	6	5	5
CFS	7	7	5	5

Similarly, the simulation results, for LFM and CFS signals, are shown in Figs. 3.14 and 3.15, respectively. Reasonable result is obtained by taking $Q = 6$ for $\alpha = 0.98$, in the case of both LFM and CFS signals. The minimum acceptable values of parameter Q, obtained through simulation for different signals and values of α, are given in Table 3.2.

Thus, it has been seen that the distortion, more significant during the end of the resulting time domain sequence for a given α, reduces with the increase in value of Q, or otherwise the length of the zero padding. Furthermore, Q reduces as α approaches unity when ideally no zero-padding is required.

However, it can be concluded, from the above discussions, that reasonably good results are obtained irrespective of the frequency, slope of frequency and the length of the original time domain sequence in the case of implementing Doppler shift in a signal using frequency domain approach if a minimum value of Q is chosen such that $Q_{\min}\alpha = (N_{\min}/L)\alpha \approx 7$ and 5 for frequency sample mapping method and segment mapping method, respectively. However, the best outcomes are obtained for a value of $N \geq 2N_{\min}$.

3.5 Echo extension

When a pulse of signal is projected onto an underwater object, various forms of reflections occur from different parts of the object. If the dimension of the target is smaller than the sonar range resolution, it can be regarded as a point target, and if the dimension of the target is bigger than the sonar resolution it is regarded as consisting of several point targets, resulting in an elongation of the reflected pulse. The phenomenon, called pulse spreading or pulse elongation, depends on several factors such as body shape, body reflectivity, and the frequency of the incident wave.

If an object of length, l, and reflection coefficient unity throughout the length, is illuminated completely by a signal of duration t_p, the signal will be reflected from various parts of the object resulting in an increase in the duration of the reflected signal by an amount

$$t_e = \frac{2l}{c} \tag{3.24}$$

The reflected echo from a larger object can be expressed as a summation of several delayed versions of the incident signal with each delayed signal being multiplied

by an amplitude corresponding to the reflection coefficient of the part of the object from where the reflection occurs. With sampling rate f_s, and pulse duration t_p, the incident signal sequence $x(n)$ has length $L = t_p f_s$. The echo gets extended by $M = t_e f_s$ samples. Let the target length be modeled as K point reflections, with a delay of $n_d = (M\text{-}1)/K$ samples between two successive reflections. The reflected echo, thus, can be written as [12]:

$$y(n) = \sum_{m=0}^{(M-1)/n_d} h(mn_d)x(n - mn_d) \tag{3.25}$$

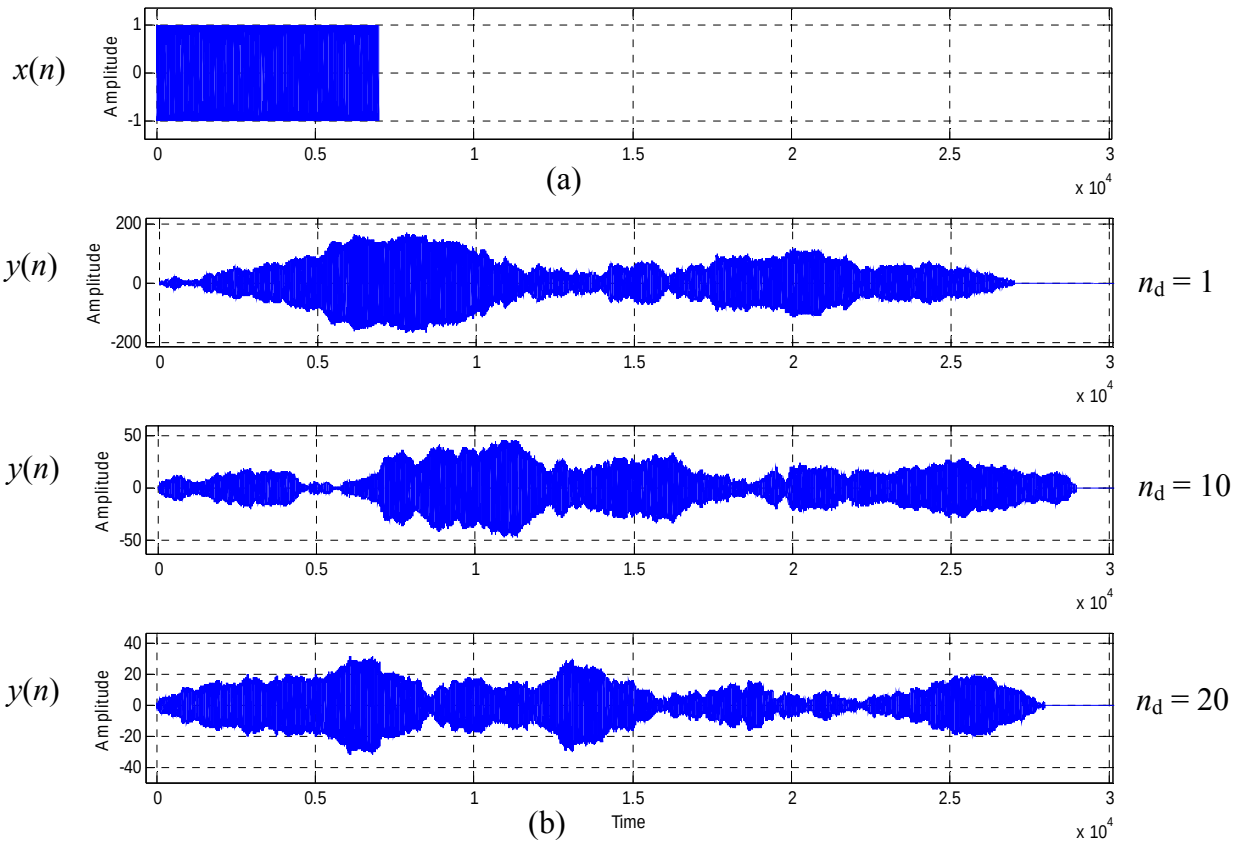


Fig. 3.16 Echo extension shown with normal distribution as target highlights: (a) PCW signal of pulse width 35 ms, and (b) Elongated echo from a 75 m target. Time axis shows the indices corresponding to f_s = 200 k sa/s.

While implementing echo extension, the entire body of the target is assumed to be divided into a number of sections and the superposition of the reflections from the centers

of these sections can be viewed as an approximation of the extension of the echo. Letting n_d = 1 results in dividing the target body into more number of sections. However, difficulty arises due to the inefficiency in computation of echo extension operation involving a long duration of the target highlights. Moreover, it is assumed that the reflections of signal from very close distances do not contribute in the extension of the echo significantly except for the amplitude of the echo, which increases considerably. Hence, the entire body of the target can be divided into fewer sections and the target highlights, of duration M, can be obtained by considering non-zero values only at regular intervals or delays of duration $n_d > 1$.

At a considerably short range, a target can be modeled to have several distribution characteristics according to the time and angle of incidence of the incident wave, as such, their expressions require the proper highlight distribution onto the target. A multi-distribution random number system has been cited in literatures for radar echo simulation environment where random numbers are drawn from uniform distribution, exponential distribution, Rayleigh distribution, and Gaussian distribution [12], [14]. In [13] a target modeling for Doppler Beam Sharpening (DBS) radar is accomplished by considering reflections from the target with Rayleigh distributed highlights. The desired nature of the target highlights, i.e., its distribution and the delay, n_d, can be either obtained through simulations for a required performance, or made available from the knowledge of the actual targets.

In this project, a set of random numbers having a normal distribution, zero mean and unit variance has been considered for the target highlights. Assuming that the set of random numbers and the incident waveform are independent sequences, it is not necessary to generate the object highlights each time a pulse elongation is required to be implemented unless the incident angle and the environment change with time.

Echo extension with target highlights corresponding to normal distribution is shown in Fig. 3.16. The signal considered for echo elongation is a CW signal with pulse width of t_p = 35 ms and frequency 40 kHz as shown in Fig. 3.16(a). Assuming that this signal is being received by the decoy and the reflection to be incorporated on the signal is that due to a target of length l = 75 m, the extended echoes are simulated, as shown in Fig. 3.16(b), for various values of n_d. Each extended echo is simulated using the Eqn.

3.25 where h is represented by the delayed set of random numbers in the range (0,1) generated using the MATLAB function, *randn*. Each echo will have an approximate elongation of t_e = 100 ms of the received pulse width as obtained from Eqn. 3.24, and the total duration of the elongated pulse is 135 ms. In cases where $n_d > 1$ and $(M\text{-}1)/n_d$ is not an integer, the extended echo will be less than that obtained for $n_d = 1$.

Chapter 4

SIMULATION OF PULSE DETECTION AND ECHO SYNTHESIS

This chapter provides the implementations as well as a study of performance of the approaches proposed, in Chapter 3, to generate the echo signal by the decoy. The proposed method of generating echo signal by the decoy involves signal detection, and echo synthesis that includes signal enhancement, Doppler shifting and echo extension.

Signal detection is a continuous process in the decoy once it is switched on, whereas the echo synthesis takes place only after pulse detection. From the discussions in Chapter 3, it is seen that the operation of signal enhancement is performed in frequency domain, and the Doppler shifting can be implemented either in frequency domain or in time domain. Furthermore, the pulse extension, which uses a FIR structure, can be implemented either in time domain or in frequency domain with nearly similar performance. Hence, depending upon the operation performed to implement Doppler shift, the echo synthesis operation can be carried out in either of the two ways and these are described in sections 4.2 and 4.3. Since the size of the hardware as well as the computational time increases with the size of the data to be processed, a particular case is presented in this chapter where the echo synthesis is carried out with a detected signal of duration t_p = 105 ms and a maximum echo extension of t_e = 120 ms. Considering the sampling frequency, f_s = 200 k sa/s, the lengths of the input data and the target highlights, required for the echo synthesis, are obtained as L = 21000 and M = 24000, respectively. The Doppler shift is implemented with α = 0.98 corresponding to v_{max} = 15 m/s (≈ 30 knots), as obtained from Eqn. 3.10. Sections 4.2 and 4.3 implement echo extension with the delay, n_d = 1, whereas the echo synthesis described in Section 4.5 incorporates $n_d > 1$.

A burst of a LFM signal, corrupted by an additive white Gaussian noise with an SNR of 5 dB, is used for the simulation of entire echo generation process. The signal has a start frequency of 40 kHz, sweep 2 kHz and duration of 100 ms (the duration becomes 105 ms after pulse detection). Figure 4.1(a) shows the time sequence and spectrum of the

LFM signal. The Doppler shifted reference signal, corresponding to $\alpha = 0.98$, and its spectrum are shown in Fig. 4.1(b).

4.1 Pulse detection

The pulse detection operation is implemented on the input time domain sequence using short-time energy detection method with a moving window of length W as given by Eqn. 3.1. For efficient realization, this have been implemented recursively as

$$\varepsilon(n) = \varepsilon(n - 1) + [x^2(n) - x^2(n - W)]/W \qquad (4.1)$$

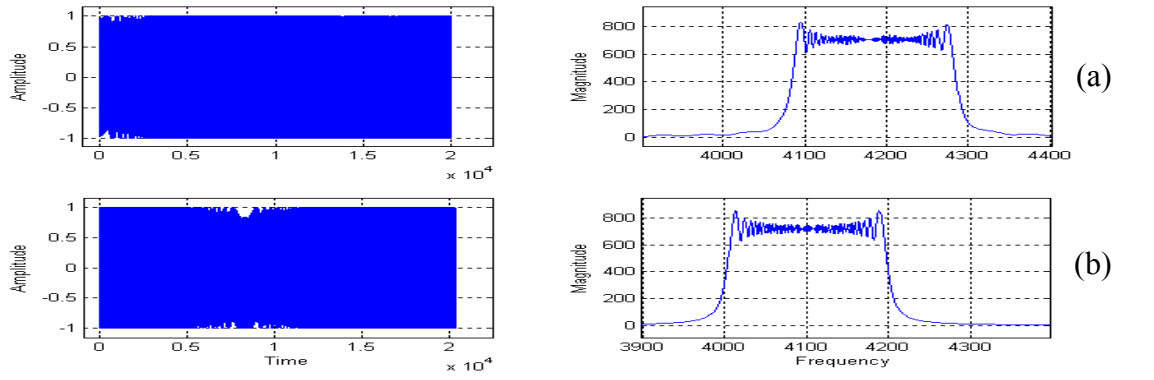


Fig. 4.1 Time domain sequences and spectra for (a) Original LFM signal, and (b) its Doppler shifted reference for $\alpha = 0.98$. Time and frequency axes show the indices corresponding to $f_s = 200$ k sa/s.

The upper threshold θ_1, and the lower threshold θ_2 are calculated once in the beginning after the decoy is launched using Eqns. 3.2 and 3.3. The decision about the presence of a pulse then consists of the following:

a) Compare $\varepsilon(n)$ with θ_1 and θ_2.

b) Declaration about signal: Present if $\varepsilon(n) > \theta_1$, and absent if $\varepsilon(n) < \theta_2$

c) Validation of signal Pulse: Valid if the width is more than 2.5 ms, else invalid.

4.2 Echo synthesis with Doppler shift implemented in time domain ($n_d = 1$)

In this approach, the Doppler shift is implemented first on the detected input sequence in time domain followed by the implementations of signal enhancement and echo extension

in the frequency domain. The actual order of echo synthesis should be that the operation of Doppler processing is performed after signal enhancement, but the computational complexity increases because the received signal has to be converted to frequency domain before the signal enhancement operation is performed and subsequently, converted back to time domain to implement Doppler shifting, and finally the Doppler shifted signal again has to be converted to frequency domain for implementing echo extension. Because of the complex conjugate symmetry in the FFT outputs, any operation on the frequency-domain output of length N needs to be carried out for the first $N/2+1$ outputs of the FFT and finally the remaining outputs are replaced by the complex conjugates of the first $N/2-1$ FFT outputs. For $\alpha = 0.98$, the length of the sequence, after Doppler shift, becomes $L' = L/\alpha \approx 21430$. Supposing that the echo synthesis is implemented on the entire length of the received data at once, the Doppler shift has to be implemented on the entire length, L, of the input data and subsequently, each of the sequences of Doppler shifted data and target highlights should be padded with zeros to make its length at least $N = L'+M-1 = 45429$ to avoid aliasing while implementing echo elongation. However, $N = 64$ k is considered the minimum required value to perform FFT algorithm on these sequences. Thus, a large amount of data processing is required resulting in a need for large memory and computational capability in the hardware. Hence, another scheme to implement the echo synthesis has been tested in which the input data sequence is split into smaller sub-sequences and each sub-sequence is operated on to generate a smaller echo. The complete echo is then generated by combining these smaller echoes using overlap-add method. The following two sub-sections describe these two methods of echo synthesis.

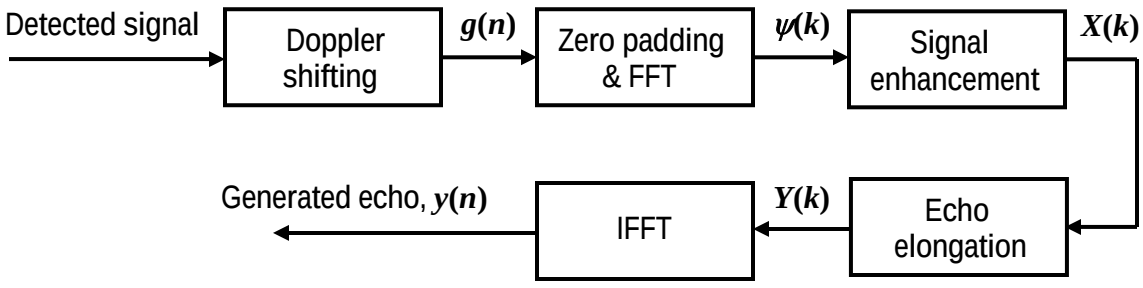


Fig. 4.2 Block diagram of echo synthesis process on a large input sequence using Doppler shifting in time domain.

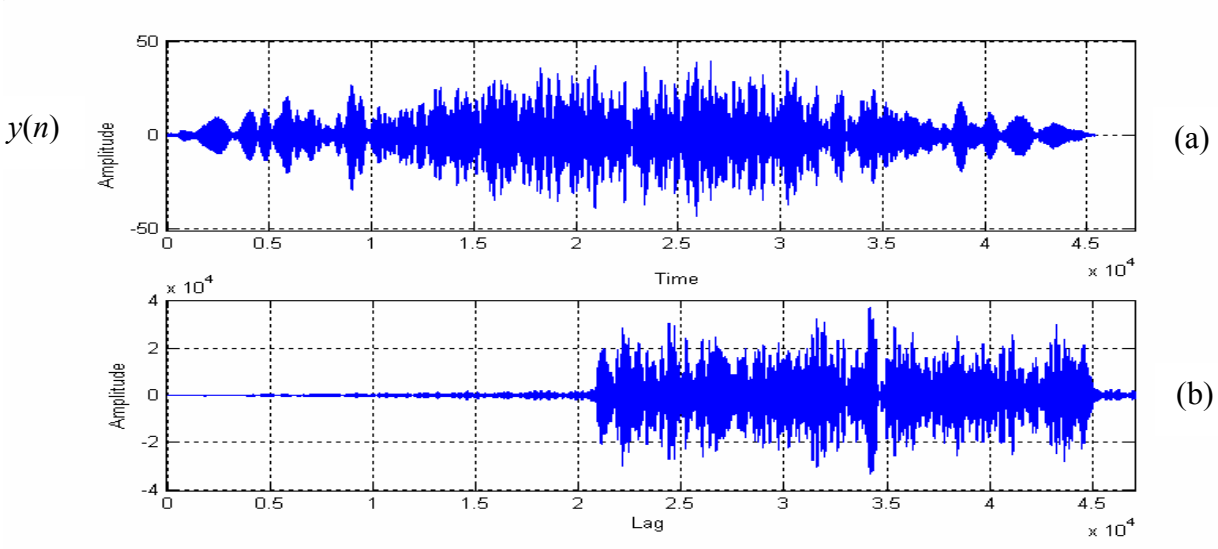


Fig. 4.3 Result of echo synthesis: (a) extended echo in time domain, and (b) the cross-correlation output. Time and lag axes show the indices corresponding to f_s = 200 k sa/s.

4.2.1 Echo synthesis using a large data sequence

The power spectrum of the noise, $P_n(k)$, and the FFT output of the zero-padded target highlights, $H(k)$, are calculated once in the beginning after the decoy is launched and the values for $k = 0, 1, \ldots, N/2$ are stored in memory. The echo synthesis then consists of the following steps

1) Using Eqns. 3.22-3.23, implement Doppler shift on the time domain signal sequence obtained after detection.
2) Obtain $\psi(k)$ by performing FFT on the Doppler shifted time sequence padded with zeros to a length N.
3) Find $P_{s+n}(k) = |\psi(k)|^2$, $\theta(k) = \text{phase}(\psi(k))$ for $k = 0, 1, \ldots, N/2$ and get enhanced signal spectrum, $X(k)$ for $k = 0, 1, \ldots, N/2$, using Eqn. 3.6-3.8.
4) Implement $Y(k) = X(k)H(k)$ for $k = 0, 1, \ldots, N/2$.
5) $Y(k)$ for $k = N/2+1, \ldots, N-1$ are obtained by taking the complex conjugates of $Y(k)$ for $k = N/2-1, \ldots, 1$.
6) The real part of the IFFT of $Y(k)$ is the desired simulated echo in time-domain.

The steps of the echo synthesis process, as mentioned above, are shown in Fig. 4.2. The simulation results after the echo synthesis are shown in Fig. 4.3. The cross-

correlation result of the extended echo with the Doppler shifted reference is shown in Fig. 4.3(b). The computational complexity and the amount of storage involved during the implementation of this approach are shown in Table 4.1. Each output point of the Doppler processing requires 3 real multiplications and 3 real additions. Each output sample of the signal enhancement routine involves 2 real multiplications and 1 real addition. Each complex multiplication involves 4 real multiplications and 2 real additions. Thus, assuming L = 21000 and N = 64 k, the total computational complexity involves 4455208 real multiplications and 6454053 real additions. The total amount of storage involved is 250379.

4.2.2 Echo synthesis using overlap-add method

In this method, the operation of echo synthesis is implemented considering smaller blocks of input data arriving at the decoy. Although the operations of Doppler shifting and signal enhancement can be performed on each independent block of input data, the operation of echo elongation involves the convolution between the entire sequences of both the input data and the target highlights, $h(n)$. Hence, the technique of implementing the convolution operation between two larger sequences, $x(n)$ and $h(n)$, using smaller sub-sequences, is discussed below in details.

Table 4.1 Computational complexity and storage involved during echo synthesis using a large data sequence (Doppler shift in time domain).

Steps	Number of real multiplications	Number of real additions	Storage
1. Doppler shift ($L^{/}$-point)	$3L^{/}$	$3L^{/}$	L
2. N-point FFT	$4(N/2)\log_2 N$	$6(N/2)\log_2 N$	$2N$
3. Signal enhancement ($N/2+1$)-point	$2(N/2+1)$	$(N/2+1)$	$(N/2+1)$ for $P_n(k)$
4. ($N/2+1$) complex multiplication	$4(N/2+1)$	$2(N/2+1)$	$2(N/2+1)$ for $H(k)$
6. N-point IFFT	$4(N/2)\log_2 N$	$6(N/2)\log_2 N$	-----

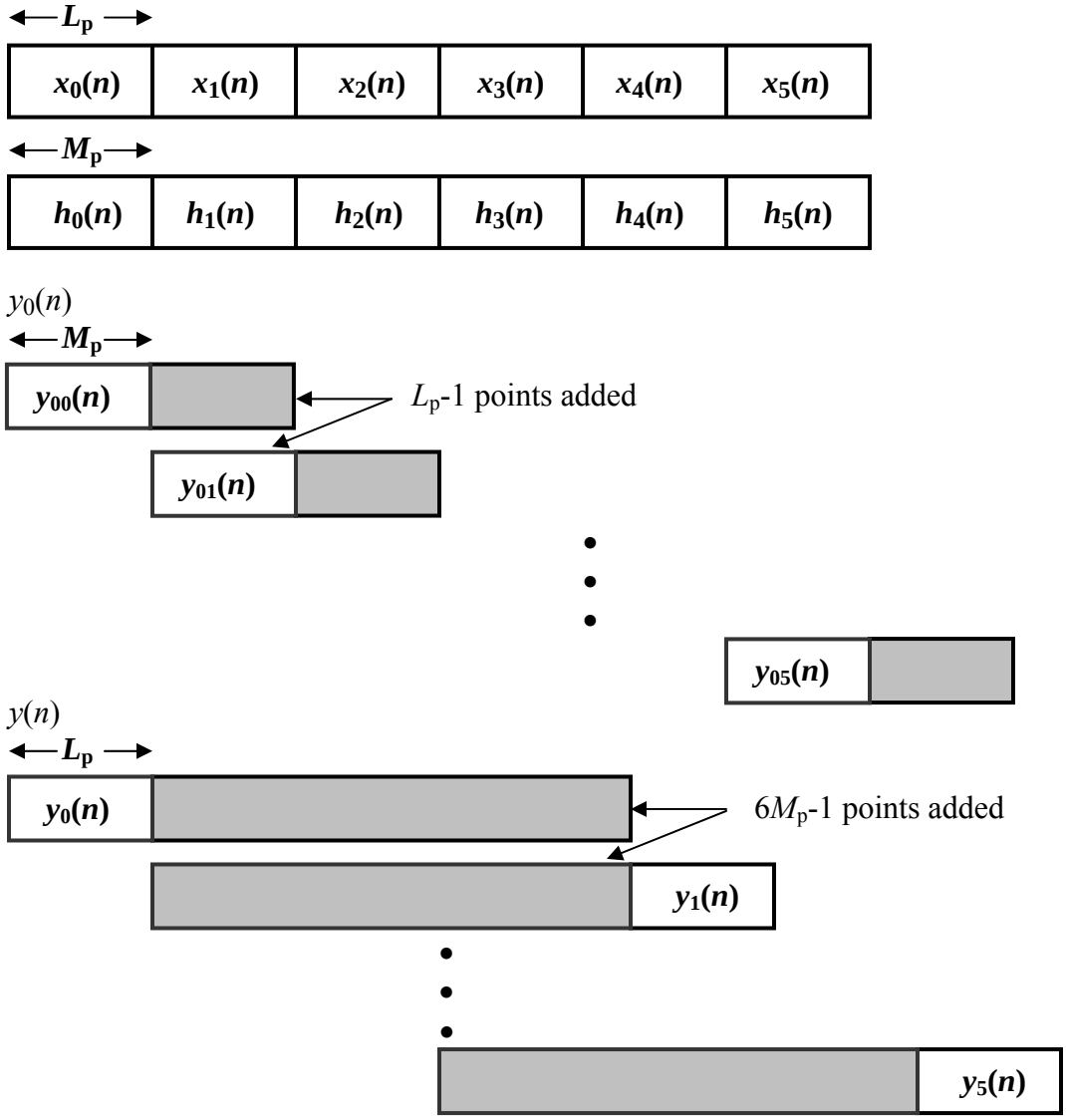


Fig. 4.4 Linear convolution using overlap-add method.

In this method, both $x(n)$ and $h(n)$ are segmented to blocks of smaller size prior to processing. Each smaller block of $x(n)$ is processed via DFT and IDFT with all the smaller blocks of $h(n)$ and the output blocks are fitted together to form the overall output sequence. For the size of the input data block as L_p points and the block of target highlights having duration M_p, the size of the DFT and IDFT is $N_p = L_p + M_p - 1$. We can assume that both the sequences, $x(n)$ and $h(n)$, have equal lengths and $L_p = M_p$ without loss of generality. We compute the N_p-point DFTs by appending $L_p - 1$ zeros to each block of $x(n)$ and $h(n)$. The two N_p-point DFTs are multiplied together to form

$$Y_{u,v}(k) = X_u(k)H_v(k) \qquad k = 0, 1, \ldots, N_p - 1 \qquad (4.2)$$

The IDFTs

$$y_{u,v}(n) = \text{IDFT}\{Y_{u,v}(k)\} \qquad (4.3)$$

yield data blocks of length N_p that are free of aliasing. Since each block of input data and target highlights is terminated with $L_p - 1$ zeros, the last $L_p - 1$ points from each output block, $y_{u,v}(n)$, must be overlapped and added to the first $L_p - 1$ points of the succeeding block to yield $y_u(n)$. The final output sequence is obtained by overlapping and adding the sequences $y_u(n)$ as illustrated in Fig. 4.4.

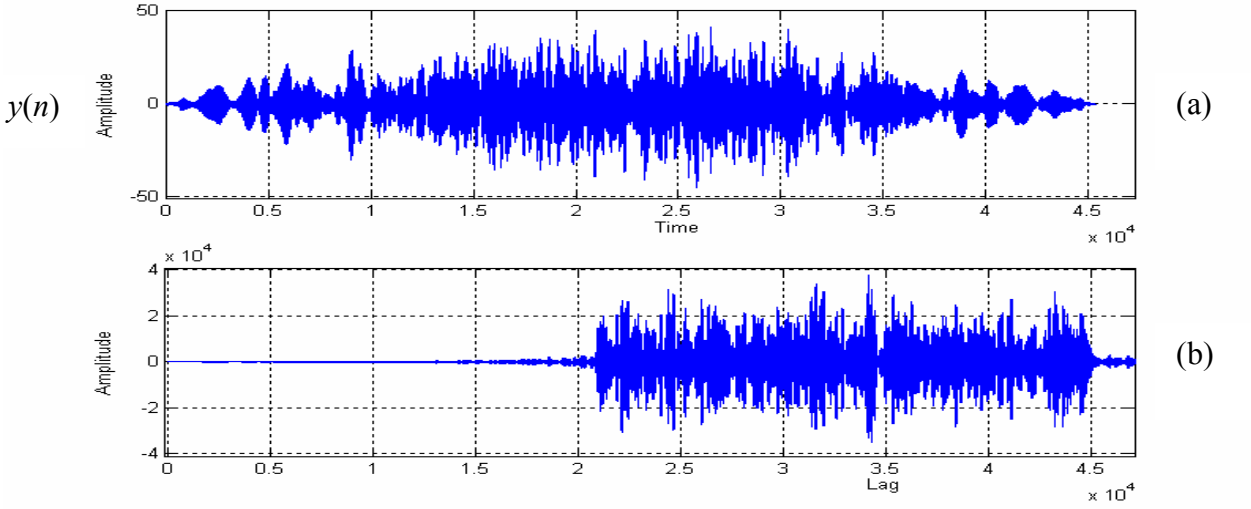


Fig. 4.5 Result of echo synthesis: (a) extended echo in time domain, and (b) Cross-correlation output. Time and lag axes show the indices corresponding to f_s = 200 k sa/s.

Considering N_p = 8 k, we choose $L_p = M_p$ = 4 k and the maximum number of blocks for each of $x(n)$ and $h(n)$ can be obtained as $\max(M, L')/L_p \approx 6$. The block of received input data for the echo synthesis thus has a duration $L_s = L_p\alpha = 4014$. The power spectrum of the noise, $P_n(k)$, is calculated once in the beginning after the decoy is launched and the values for $k = 0, 1, \ldots, N_p/2$ are stored in memory. The steps for echo synthesis thus consist of

1) Using Eqns. 3.22-3.23, implement Doppler shift on the input time domain signal sequence of duration L_s obtained after detection.

2) Obtain $\psi_u(k)$ by performing FFT on the Doppler shifted time sequence padded with zeros to a length N_p.
3) Find $P_{s+n}(k) = |\psi_u(k)|^2$, $\theta(k)$ = phase($\psi_u(k)$) for k = 0, 1, …, $N_p/2$ and get enhanced signal spectrum, $X_u(k)$ for k = 0, 1, …, $N_p/2$, using Eqn. 3.6-3.8.
4) Obtain $H_v(k)$ for k = 0, 1, …, $N_p/2$.
5) Obtain $y_u(n)$ as shown in Fig. 4.4.
6) Repeat Steps 1 to 5 for the next block of input data and the final data sequence is obtained as shown in Fig. 4.4.

The simulated extended echo obtained through this synthesis approach is shown in Fig. 4.5. The cross-correlation output of the generated echo with the Doppler shifted reference is shown in Fig. 4.5(b). The computational complexity and the amount of storage involved during the implementation of this approach are shown in Table 4.2.

Table 4.2 Computational complexity and storage involved during echo synthesis using overlap-add method (Doppler shift in time domain).

Steps	Number of real multiplications	Number of real additions	Storage
1. Doppler shift (L_p-point)	$3L_p$	$3L_p$	L_s M for $h(n)$
2. N_p-point FFT	$4(N_p/2)\log_2 N_p$	$6(N_p/2)\log_2 N_p$	$2N_p$
3. Signal enhancement ($N_p/2+1$)-point	$2(N_p/2+1)$	$(N_p/2+1)$	$(N_p/2+1)$ for $P_n(k)$
4. Six N_p-point FFT	$6*4(N_p/2)\log_2 N_p$	$6*6(N_p/2)\log_2 N_p$	$2N_p$
5. Six ($N_p/2+1$) complex multiplication	$6*4(N_p/2+1)$	$6*2(N_p/2+1)$	-----
Six N_p-point IFFT	$6*4(N_p/2)\log_2 N_p$	$6*6(N_p/2)\log_2 N_p$	-----
Overlap-add		$5(L_p-1)$	-----
6. Overlap-add		$5(6L_p-1)$	$L'+M-1$ for output

Step 4 requires only $2N_p$ storage locations because FFTs are done sequentially. Since the entire input data is divided into 6 blocks, the Steps 1 to 5 are repeated 6 times. Assuming L_p = 4 k, L_s = 4014, and N_p = 8 k, the total computational complexity is 17867040 real multiplications and 25559083 real additions. The total amount of storage involved is 110307 locations. However, as can be seen from Fig. 4.4, the sequence $\{y(0), y(1), \ldots, y(L_p\text{-}1)\}$ is available for outputting before the second pass of repeating the Steps from 1 to 6. The sequence $\{y(L_p), y(L_p\text{+}1), \ldots, y(2L_p\text{-}1)\}$ is available for outputting before the third pass of repeating the steps from 1 to 6, and so on. Thus, the computational complexity for each output block of length L_p involves 2977840 real multiplications and 4263943 real additions. A further reduction in the computational complexity in Table 4.2 can be achieved by eliminating the FFT operations that are performed for the target highlights in Step 4. Thus, instead of computing the FFTs of blocks of target highlights each time for a block of input data, the FFT outputs are pre-computed and stored in the memory. Since only $(N_p/2+1)$ output points of the FFT are stored, the memory required in Table 4.2 is $6*2(N_p/2+1) = 49164$. However, the requirement for M locations for $h(n)$ in Step 1 and $2N_p$ locations in step 4 are eliminated. Thus, the effective computational complexity for each output block of length L_p involves 1699888 real multiplications and 2347015 real additions. The storage requirement is 119087 locations.

4.3 Echo synthesis with Doppler shift implemented in frequency domain (n_d = 1)

The basic difference between this approach of echo synthesis and that described in Section 4.2 lies in the manner the operation of Doppler shift is implemented. The entire synthesis operations, in this approach, are carried out in frequency domain. However, as mentioned in Section 3.4, the detected time-domain input sequence needs to be padded with zeros of appropriate length before implementing the Doppler shift in frequency domain. Assuming that the Doppler shift is implemented using segment mapping method and $Q\alpha = 5$, we obtain $Q = 5.1$ corresponding to $\alpha = 0.98$. Thus, the minimum length of the zero-padded input sequence, considering the entire length of the received data, required for Doppler processing is obtained as $N = LQ = 107100$, or at least 128 k to perform a FFT operation on this sequence of input data. The sequence, $h(n)$, of the target highlights has a length much less than N and hence, the sequence is padded with zeros to

make its length N. Since $N > L'+M-1$ ($L' = L/\alpha = 21430$ and $M = 24000$), the extent of zero-padding is sufficient to avoid aliasing when echo extension is implemented in frequency domain. However, as described in Section 4.2, this method also requires a large amount of data processing if the echo synthesis is implemented on the entire length of the received data at once and hence, another scheme same as that described in Section 4.2.2 has also been investigated.

4.3.1 Echo synthesis on a large data sequence

The power spectrum of the noise, $P_n(k)$, and the FFT output of the zero-padded target highlights, $H(k)$, are calculated once in the beginning after the decoy is launched and the values for $k = 0, 1, \ldots, N/2$ are stored in memory. The echo synthesis then consists of the following steps

1) Perform FFT on the detected input sequence zero-padded to a length N.
2) The signal spectrum is frequency shifted using Eqns. 3.16-3.17 or 3.18-3.20 to get $\psi(k)$ for $k = 0, 1, \ldots, N/2$.
3) Find $P_{s+n}(k) = |\psi(k)|^2$, $\theta(k) = \text{phase}(\psi(k))$ for $k = 0, 1, \ldots, N/2$ and get enhanced signal spectrum, $X(k)$ for $k = 0, 1, \ldots, N/2$, using Eqns. 3.6-3.8.
4) Implement $Y(k) = X(k)H(k)$ for $k = 0, 1, \ldots, N/2$.
5) $Y(k)$ for $k = N/2+1, \ldots, N-1$ are obtained by taking the complex conjugates of $Y(k)$ for $k = N/2-1, \ldots, 1$.
6) The real part of the IFFT of $Y(k)$ is the desired simulated echo in time-domain.

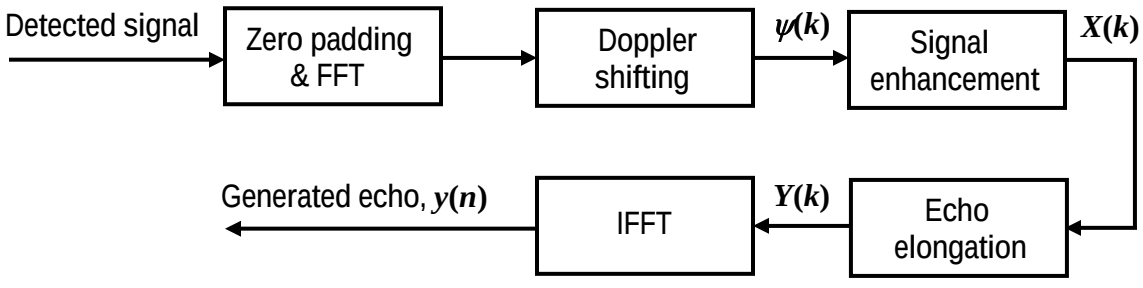


Fig. 4.6 Block diagram of the echo synthesis process on a large input sequence using Doppler shifting in frequency domain.

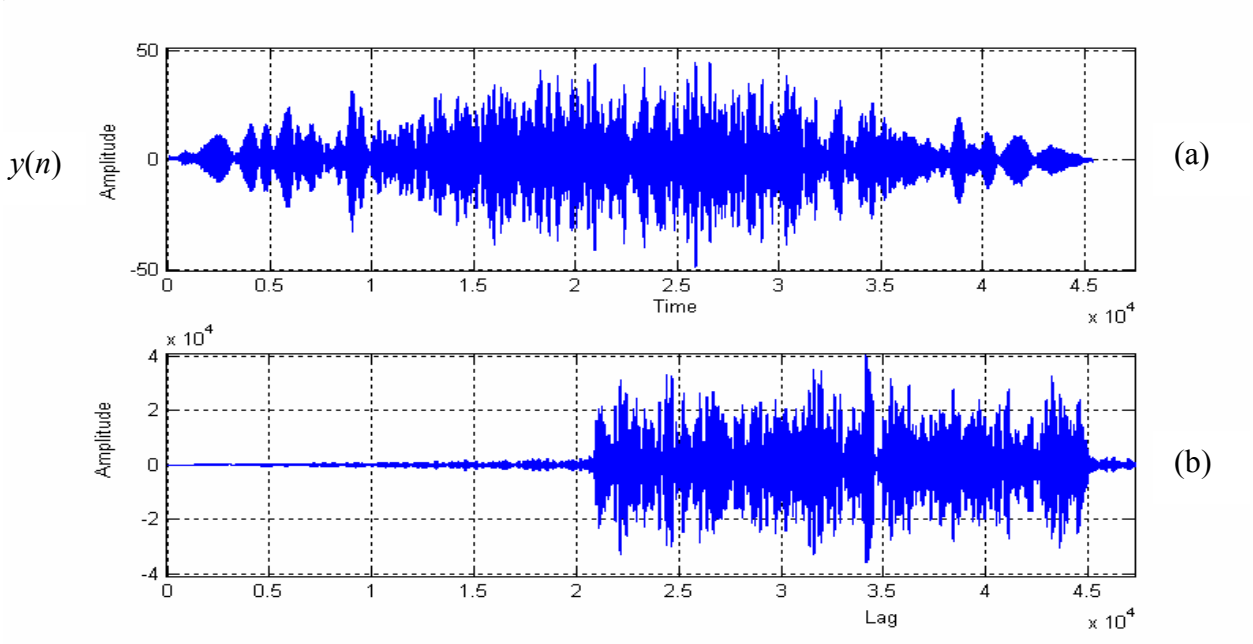


Fig. 4.7 Result of echo synthesis: (a) extended echo in time domain, and (b) Cross-correlation output. Time and lag axes show the indices corresponding to f_s = 200 k sa/s.

Table 4.3 Computational complexity and storage involved during echo synthesis using a large data sequence (Doppler shift in frequency domain).

Steps	Number of real multiplications	Number of real additions	Storage
1. N-point FFT	$4(N/2)\log_2 N$	$6(N/2)\log_2 N$	L, $2N$
2. Doppler shift (N/2+1)-point	$5(N/2+1)$	$8(N/2+1)$	$2(N/2+1)$
3. Signal enhancement (N/2+1)-point	$2(N/2+1)$	$(N/2+1)$	$(N/2+1)$ for $P_n(k)$
4. (N/2+1) complex multiplication	$4(N/2+1)$	$2(N/2+1)$	$2(N/2+1)$ for $H(k)$
5. N-point IFFT	$4(N/2)\log_2 N$	$6(N/2)\log_2 N$	-----

The steps of the echo synthesis process, as mentioned above, are shown in Fig. 4.6. The simulated extended echo and its cross-correlation result with the Doppler shifted

reference are shown in Fig. 4.7. The computational complexity and the amount of storage involved during the implementation of this approach are given in Table 4.3. The operation of Doppler shift in frequency domain involves a variable amount of computational complexity depending upon the factor, α, by which the frequency shift of the spectrum of the signal is accomplished. However, the minimum required computations are obtained as 5 real multiplications and 8 real additions corresponding to each sample of the image spectrum. Thus, assuming N = 128 k, the total computational complexity involves 9633803 real multiplications and 14090251 real additions. The total amount of storage involved is 610829 locations.

4.3.2 Echo synthesis by overlap-add method

In this method, the echo synthesis is carried out by segmenting the input data and $h(n)$ to blocks of smaller size as indicated in Section 4.2.2. Considering N_p = 8 k and Q = 5.1, the input block of data for the echo synthesis is obtained as $L_s = N_p/Q$ = 1606 and hence, $L_p = L_s/\alpha$ = 1639 and $M_p = N_p - L_p + 1$ = 6554. With these values of L_p and M_p, we have $L/L_s \approx$ 14 blocks of input data and $M/M_p \approx 4$ blocks of target highlights. The segmentation of the data sequences and the fitting of the output blocks to implement the linear convolution operation are shown in Fig. 4.8. The power spectrum of the noise, $P_n(k)$, is calculated once in the beginning after the decoy is launched and the values for $k = 0, 1, \ldots, N_p/2$ are stored in memory. The echo synthesis then consists of the following steps

1) Perform FFT on the detected input sequence of length L_s zero-padded to a length N_p.
2) The signal spectrum is frequency shifted using Eqns. 3.16-3.17 or 3.18-3.20 to get $\psi_u(k)$ for $k = 0, 1, \ldots, N_p/2$.
3) Find $P_{s+n}(k) = |\psi_u(k)|^2$, $\theta(k)$ = phase($\psi_u(k)$) for $k = 0, 1, \ldots, N_p/2$ and get enhanced signal spectrum, $X_u(k)$ for $k = 0, 1, \ldots, N_p/2$, using Eqns. 3.6-3.8.
4) Obtain $H_v(k)$ for $k = 0, 1, \ldots, N_p/2$
5) Obtain $y_u(n)$ as shown in Fig. 4.8.
6) Repeat Steps 1 to 5 for the next block of input data and the final data sequence is obtained as shown in Fig. 4.8.

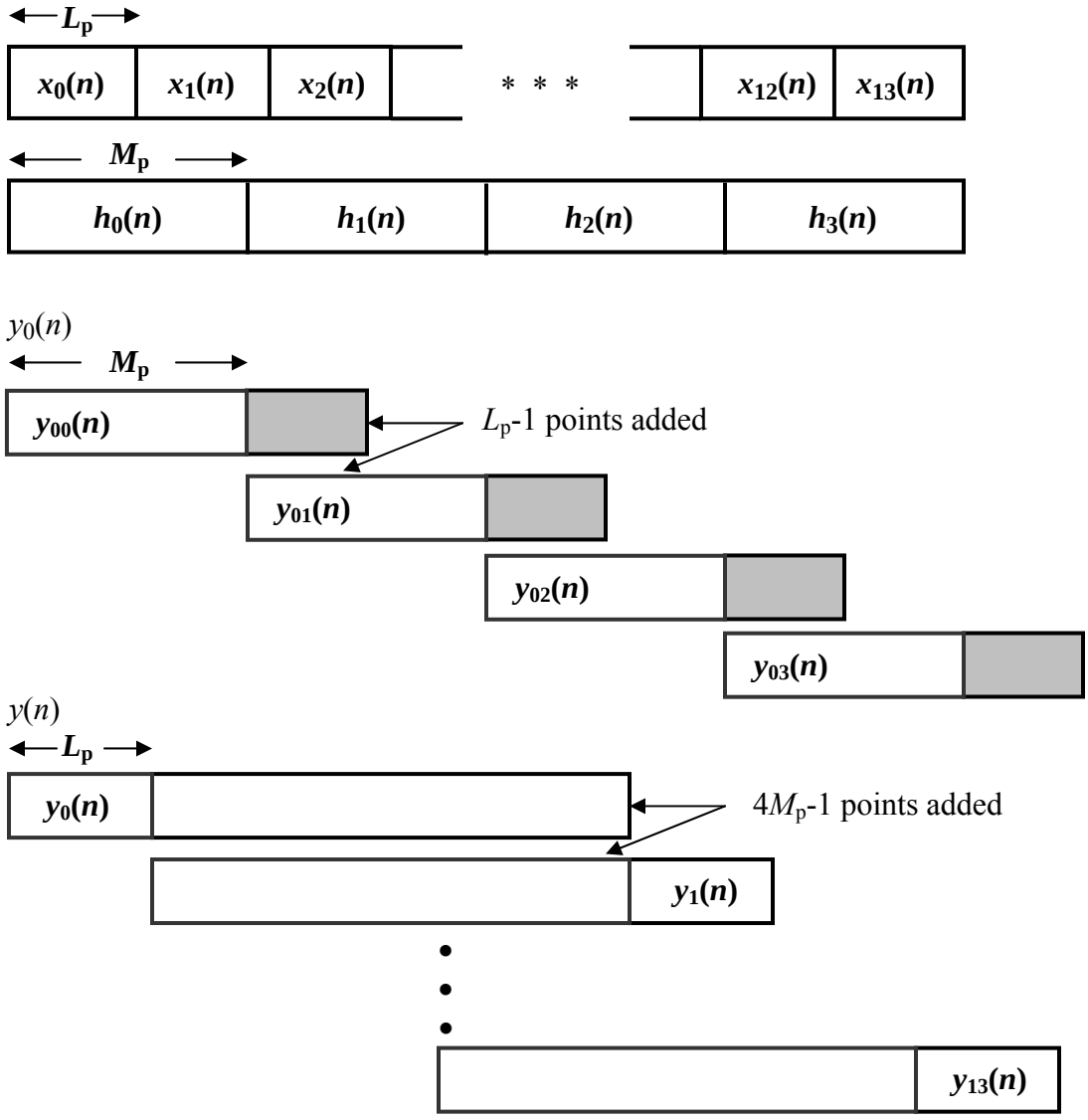


Fig. 4.8 Linear convolution using overlap-add method.

The simulated extended echo and its cross-correlation result with the Doppler shifted reference are shown in Fig. 4.9. The computational complexity and the amount of storage involved during the implementation of this approach are given in Table 4.4. Since, the entire input data is divided into 14 blocks, the Steps 1 to 5 are repeated 14 times. Assuming N_p = 8 k, L_s = 1606, and L_p = 1639, M_p = 6554, total computational complexity involves 28156226 real multiplications and 41640165 real additions. The total amount of storage involved is 116093 locations. However, as seen from Fig. 4.8, the output sequence $\{y(0), y(1), \ldots, y(L_p\text{-}1)\}$ is available for outputting before the second

pass of repeating the Steps 1 to 5. Similarly, the output sequence $\{y(L_p), y(L_p+1), \ldots, y(2L_p-1)\}$ is available for outputting before the third pass of repeating the Steps 1 to 5. Thus, the computational complexity for each output block of length L_p involves 2011159 real multiplications and 2976170 real additions.

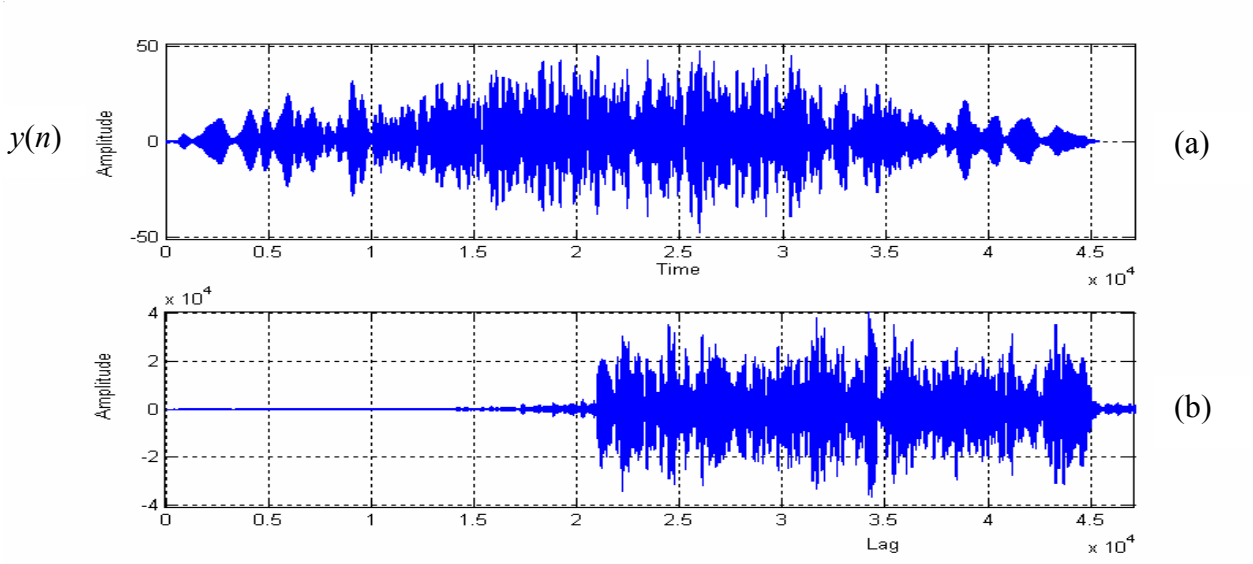


Fig. 4.9 Result of echo synthesis: (a) extended echo in time domain, and (b) Cross-correlation output. Time and lag axes show the indices corresponding to f_s = 200 k sa/s.

As mentioned in Section 4.2.2, the computational complexities in Table 4.4 can be further reduced by eliminating the FFT operations that are performed for the target highlights in Step 4 and storing the FFT outputs of the zero-padded blocks of target highlights in the memory. Since only $(N_p/2+1)$ output points of the FFT are stored, the memory required in Table 4.4 is $4*2(N_p/2+1) = 32776$ whereas the requirement of M locations for $h(n)$ in Step 1 and $2N_p$ locations in Step 4 are eliminated. Thus, the effective computational complexity for each output block of length L_p involves 1159191 real multiplications and 1698218 real additions. The storage involved is 108485 locations.

4.4 Comparisons of the echo synthesis methods (n_d = 1)

From the simulation results of sections 4.2 and 4.3, it is seen that the synthesized echoes are same irrespective of the methods proposed in those sections except for the computational and hardware complexities. The summary of the computational

complexity and amount of storage involved, in these approaches of echo synthesis process, are given in Table 4.5. In view of implementing the echo synthesis process, it is clear that the approaches, described in Sections 4.2.1 and 4.3.1, involve large amount of storage as well as the delay in producing the output. The large storage is caused due to the fact that large sequences of data, zero-padded to implement echo extension, are processed at once. The approach, using Doppler shift in frequency domain, requires the maximum storage because of the large amount of zero-padding that is required in the input data to avoid distortion while implementing Doppler shift. The delay occurs because the output is generated only after collecting and processing the entire input data.

Table 4.4 Computational complexity and storage involved during echo synthesis using overlap-add method (Doppler shift in frequency domain).

Steps	Number of real multiplications	Number of real additions	Storage
1. N_p-point FFT	$4(N_p/2)\log_2 N_p$	$6(N_p/2)\log_2 N_p$	L_s, $2N_p$, M for $h(n)$
2. Doppler shift ($N_p/2+1$) point	$5(N_p/2+1)$	$8(N_p/2+1)$	$2(N_p/2+1)$
3. Signal enhancement ($N_p/2+1$)-point	$2(N_p/2+1)$	$(N_p/2+1)$	$(N_p/2+1)$ for $P_n(k)$
4. Four N_p-point FFT	$4*4(N_p/2)\log_2 N_p$	$4*6(N_p/2)\log_2 N_p$	$2N_p$
5. Four ($N_p/2+1$)comp. multiplication	$4*4(N_p/2+1)$	$4*2(N_p/2+1)$	-----
Four N_p-point IFFT	$4*4(N_p/2)\log_2 N_p$	$4*6(N_p/2)\log_2 N_p$	-----
Overlap-add		$3(L_p-1)$	-----
6. Overlap-add		$13(4M_p-1)$	$L'+M-1$ for output

The overlap-add methods of implementing the echo synthesis, on the other hand, require less amount of storage and generates output in blocks thus introducing less delay in outputting the required echo. However, the overlap-add method, using Doppler shift in

time domain as described in Section 4.2.2, has been found most efficient computationally. From Table 4.5, the computational efficiency of Approach 2 over that of Approach 4 is obtained as

$$\eta = \frac{\text{Output block of Approach 2}}{\text{Output block of Approach 4}} \times \frac{(\gamma_m + \gamma_a) \text{ of Approach 4}}{(\gamma_m + \gamma_a) \text{ of Approach 2}}$$

$$\approx 1.8$$

i.e., performing the same number of arithmetic operations, Approach 2 can process almost twice the length of the input data that can be processed by the Approach 4.

Table 4.5 Summary of computational complexity and storage.

Approach	Output block length	Number of real multiplications, Υ_m	Number of real additions, Υ_a	Storage
1. Section 4.2.1	45428	4455208	6454053	250379
2. Section 4.2.2	4096	1699888	2347015	119087
3. Section 4.3.1	45428	9633803	14090251	610829
4. Section 4.3.2	1639	1159191	1698218	108485

4.5 Echo synthesis with $n_d > 1$

The method of echo synthesis, for $n_d > 1$, is described here using the method mentioned in Section 4.2.2 with Doppler shift implemented in time domain but only the input data is segmented into smaller blocks. Since the length of the target highlights remains same, carrying out the echo extension operation in frequency domain does not offer any advantage over the methods discussed before and hence, the echo extension operation is implemented in time domain. Assuming that L_p = 4 k, the segmentation of the input data and the generation of the extended echo are shown in Fig. 4.10. The method of implementing the partially extended echo, $y_0(n)$, corresponding to the input data block $x_0(n)$, is shown in Fig. 4.11.

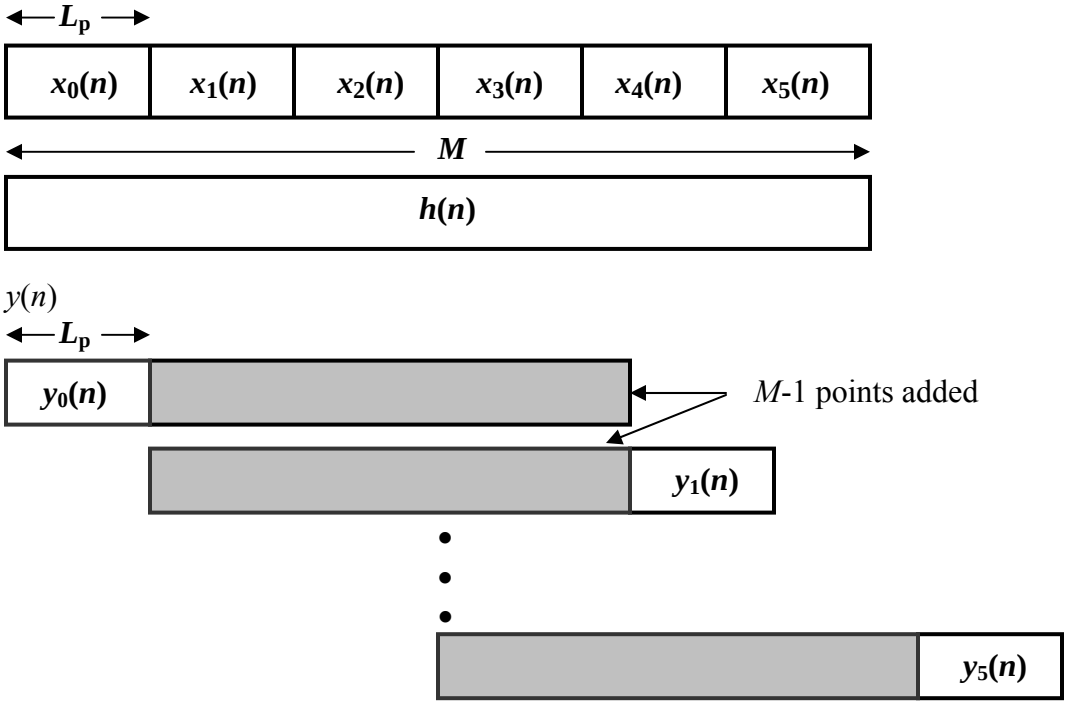


Fig. 4.10 Linear convolution using overlap-add method.

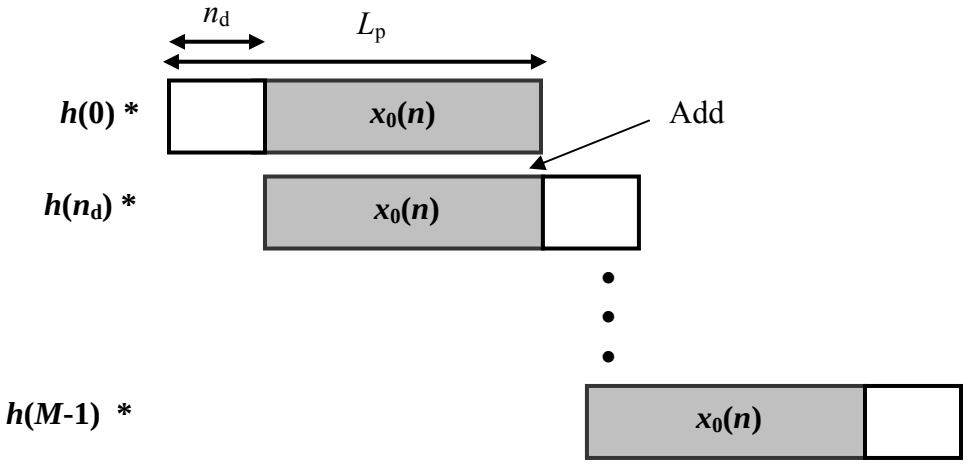


Fig. 4.11 Partially extended echo, $y_u(n)$.

Assuming a zero-initialized destination buffer of length $L'+M-1$, each row of Fig. 4.11 involves L_p real multiplications and L_p real additions. Since there are $(M-1)/n_d+1$ rows, the total computations involve $[(M-1)/n_d+1]L_p$ real multiplications and $[(M-1)/n_d+1]L_p$ real additions. The same computational complexity is involved to generate and add

$y_1(n)$ to $y_0(n)$, and so on. In other words, each output block of length L_p requires $[(M\text{-}1)/n_d + 1]L_p$ real multiplications and $[(M\text{-}1)/n_d + 1]L_p$ real additions. The steps for echo synthesis for a single input block, thus, consists of

1) Using Eqn. 3.22-3.23, implement Doppler shift on the input time domain signal sequence of duration L_s obtained after detection.
2) Obtain $\Psi_u(k)$ by performing FFT on the Doppler shifted time sequence of length L_p (no zero-padding is required in the time domain sequence).
3) Find $P_{s+n}(k) = |\Psi_u(k)|^2$, $\theta(k)$ = phase($\Psi_u(k)$) for k = 0, 1, …, $L_p/2$ and get enhanced signal spectrum, $X_u(k)$ for k = 0, 1, …, $L_p/2$, using Eqns. 3.6-3.8.
4) Obtain real part of the IFFT of $X_u(k)$ for k = 0, 1, …, L_p-1.
5) Obtain $y_u(n)$ as shown in Fig. 4.11.

The complete echo is generated by repeating the above steps six times. The computational complexity and the storage requirement to generate the echo $y_u(n)$ are given in Table 4.6.

Table 4.6 Computational complexity and storage involved during the synthesis of an echo of length L_p ($n_d > 1$).

Steps	Number of real multiplications	Number of real additions	Storage
1. Doppler shift L_p-point	$3L_p$	$3L_p$	L_s M for $h(n)$
2. L_p-point FFT	$4(L_p/2)\log_2 L_p$	$6(L_p/2)\log_2 L_p$	$2L_p$
3. Signal enhancement ($L_p/2$+1)-point	$2(L_p/2+1)$	$(L_p/2+1)$	$(L_p/2+1)$ for $P_n(k)$
4. L_p-point IFFT	$4(L_p/2)\log_2 L_p$	$6(L_p/2)\log_2 L_p$	-----
5. Overlap-add	$[(M\text{-}1)/n_d + 1]L_p$	$[(M\text{-}1)/n_d + 1]L_p$	$L' + M$-1

With n_d = 20, the total computational complexity, for L_s = 4014 input points, involves 5132289 real multiplications and 5228545 real additions. The total storage

required is 83683 locations. It is seen that the computational complexity has an inverse relationship with the value of n_d and an increase in computational efficiency results only with a large value of n_d.

4.6 Signal enhancement applied to underwater torpedo recording

In order to better understand the characteristics of the underwater acoustic signal and application of signal enhancement technique (Section 3.2), recording of echo received by a torpedo in an underwater trial was used. The trial was conducted in the Nagarjunasagar lake near Hyderabad with the surface ship of length 50 m, and the torpedo being operated in active mode at a depth of 40 m. The speeds of the torpedo and the ship were 12.5 m/s (≈ 25 knots) and 7.5 m/s (≈ 15 knots), respectively. The input signal to the torpedo is passed through the preamplifier, band-pass filter, time varying gain (TVG) circuit, and finally a 12-bit ADC, the output of which is recorded for a total duration of 6 min. The data are later transferred to a PC for off-line processing.

The data received by the torpedo consists of target echo, surface and volume reverberation, and the underwater noise that includes the flow noise during the trial. The reverberation in the received data is dominant at the short range due to the surface reflections and dies away slowly with time when the reverberation is caused mainly by the small oceanic objects. The profile of the TVG circuit is designed to invert the effect of strong reverberation at short ranges. The reverberation from the water surface introduces a small Doppler to the received signal due to the velocity, 2-3 knots, of the surface caused by wind whereas the bottom reverberation does not cause any Doppler to the received signal.

The torpedo was operated in PCW and LFM modes of transmission. Recording of the received signal in both the modes are used here. The received data, corresponding to two reception cycles of 2.7 s PRT, are processed and shown in Figs. 4.12 – 4.15. Figure 4.12(a) shows the received data when a PCW signal of pulse width 35 ms and frequency 40 kHz was transmitted.

The location of the echo is obtained by performing the cross-correlation of the received data. Since correlation receiver is sensitive to Doppler shift for PCW signal, the cross-correlation output rejects the reverberation caused by the surface. The cross-

correlation of the received sequence with the transmitted signal is shown in Fig. 4.12(b). With the Doppler shifted reference (Doppler shift introduced on the basis of knowledge of the torpedo and ship velocities), it is shown in Fig. 4.12(c). It can be seen that the cross-correlation with the transmitted pulse does not show the echo, but cross-correlation with the Doppler shifted pulse locates the echo precisely. Fig. 4.12(d) shows the amplitude spectrum of the received data in dB scale. An estimate of the mean-squared value of signal plus noise during the pulse reception and noise elsewhere gave an SNR estimate of 2 dB.

Signal enhancement based on spectral subtraction (as described in Section 3.2) was applied on the signal of Fig. 4.12(a). Spectral subtraction factor $\alpha = 2$ gave adequate noise suppression. Figure 4.13 shows the enhanced signal as well as cross-correlation results. For this enhancement, noise estimate was carried out by averaging the squared-magnitude spectra over successive placement of 35 ms windows before the pulse reception.

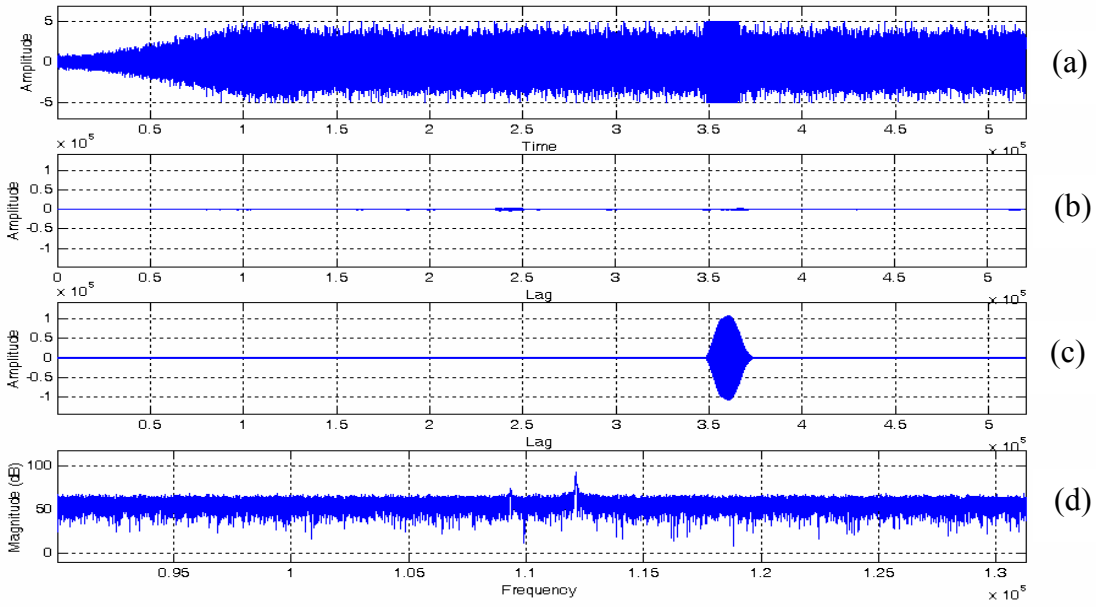


Fig. 4.12 Results of processing input data to the torpedo during dynamic trial (transmitted signal is PCW): (a) Input data, (b) Cross-correlation of input data with transmitted signal, (c) Cross-correlation of input data with Doppler shifted reference, and (d) dB spectrum of the input data. Time, lag and frequency axes show the indices corresponding to f_s = 200 k sa/s.

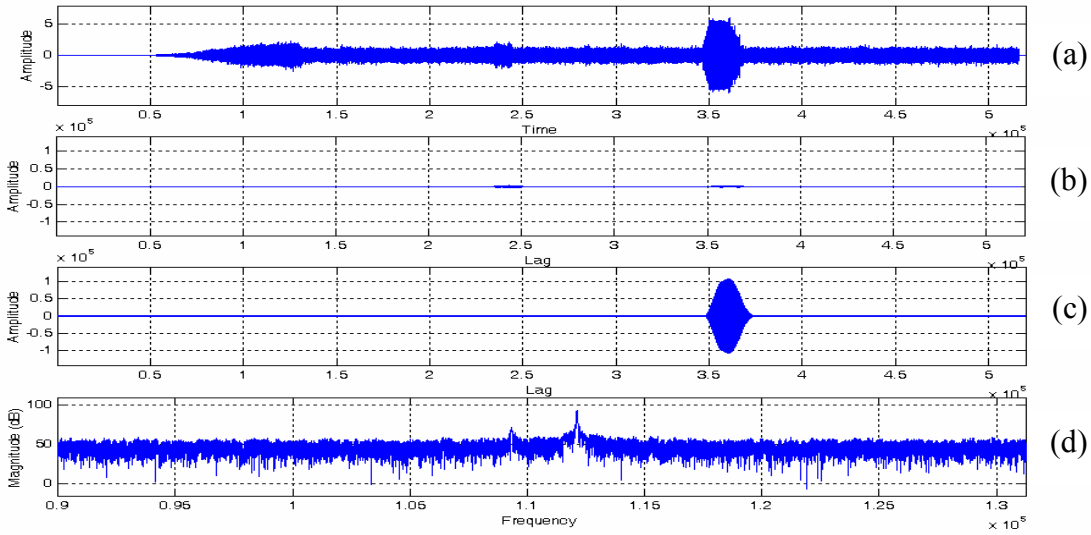


Fig. 4.13 Results of processing input data to the torpedo during dynamic trial (transmitted signal is PCW): (a) Enhanced input data, (b) Cross-correlation of the enhanced input data with transmitted signal, (c) Cross-correlation of this enhanced input data with Doppler shifted reference, and (d) dB spectrum of the enhanced input data. Time, lag and frequency axes show the indices corresponding to f_s = 200 k sa/s.

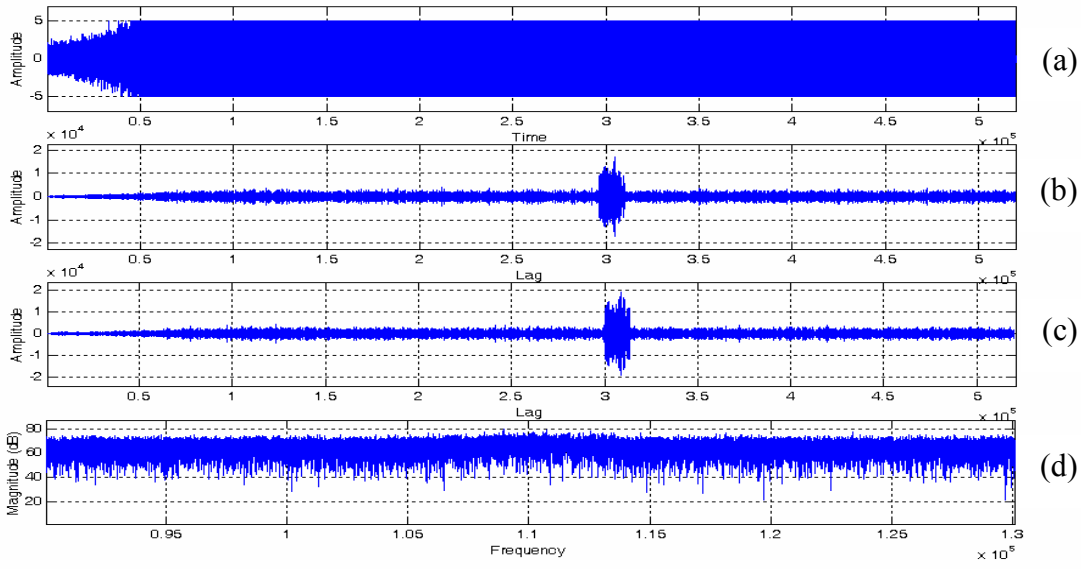


Fig. 4.14 Results of processing input data to the torpedo during dynamic trial (transmitted signal is LFM): (a) Input data, (b) Cross-correlation of input data with transmitted signal, (c) Cross-correlation of input data with Doppler shifted reference, and (d) dB spectrum of the input data. Time, lag and frequency axes show the indices corresponding to f_s = 200 k sa/s.

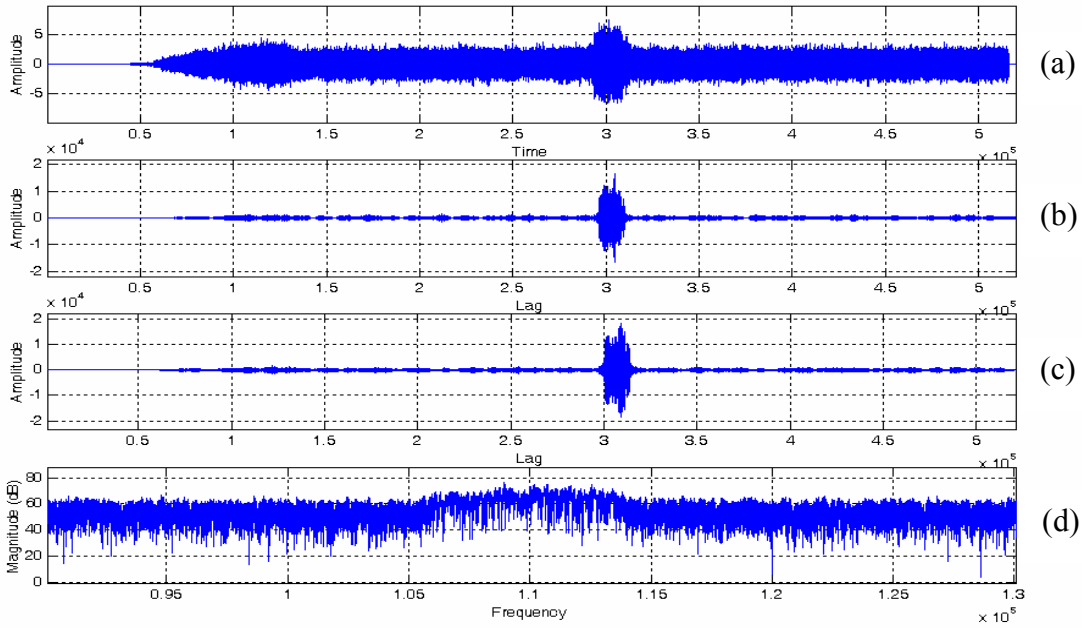


Fig. 4.15 Results of processing input data to the torpedo during dynamic trial (transmitted signal is LFM): (a) Enhanced input data, (b) Cross-correlation of the enhanced input data with transmitted signal, (c) Cross-correlation of this enhanced input data with Doppler shifted reference, and (d) dB spectrum of the enhanced input data. Time, lag and frequency axes show the indices corresponding to f_s = 200 k sa/s.

Figure 4.14(a) shows the data received by the torpedo when a LFM signal of pulse width 35 ms, start frequency 40.6 kHz, and sweep 2 kHz was transmitted. The cross-correlation results of this sequence with the transmitted signal and the Doppler shifted reference are shown in Figs. 4.14(b) and 4.14(c), respectively. The amplitude spectrum of the received data sequence in dB scale, is shown in Fig. 4.14(d). The results after enhancement of the received signal are shown in Fig. 4.15. For this signal, the SNR of the input data was estimated to be -20 dB. A subtraction factor $\alpha = 3$ was found to be more effective.

Chapter 5

REAL-TIME IMPLEMENTATION

This chapter describes the software implementation of the decoy algorithms for real-time operations on a proposed digital hardware based on the ADSP-21160 digital signal processor. The choice of the processor has been made based on its speed of operation and the presence of 4 Mb of dual-ported internal memory particularly suitable for DSP applications. The entire exercise involves receiving the digital data through an input port on interrupt, performing echo generation operations, and finally sending the digital output to an output port on interrupt. In the actual system, as shown in Fig. 2.2, the input data is received from an ADC on interrupt and the output data will be sent to a DAC on interrupt generated by the processor's timer.

The development of the software has been accomplished using a PC-based real-time integrated software development and debugging environment (IDDE) with the trade name "VisualDSP++®". The software development tool contains a preprocessor, an assembler, a C-compiler, a text editor, and a real time simulator including a multiple-window graphics for data display. The simulator provides the facility of real-time execution of the application program on a PC without requiring but by merely specifying the actual hardware on which the software is intended to run. The computation time of a software component is specified in terms of the total number of cycles being executed to perform the operation and all the ports are configured in terms of the data files. The application software has been developed in ADSP-21160 assembly language with the objective of optimizing the computation time. The source code file, developed in assembly language, has a <.asm> extension while the source code file, developed using the C-language, will have the extension of <.c>. The software development process requires a hardware description file, with extension <.ldf>, which describes the digital hardware resources utilized while implementing the algorithms through software. The

files, with <.asm> and <.ldf> extensions, are linked to create the executable file having the same filename of the application code but with the <.dxe> extension.

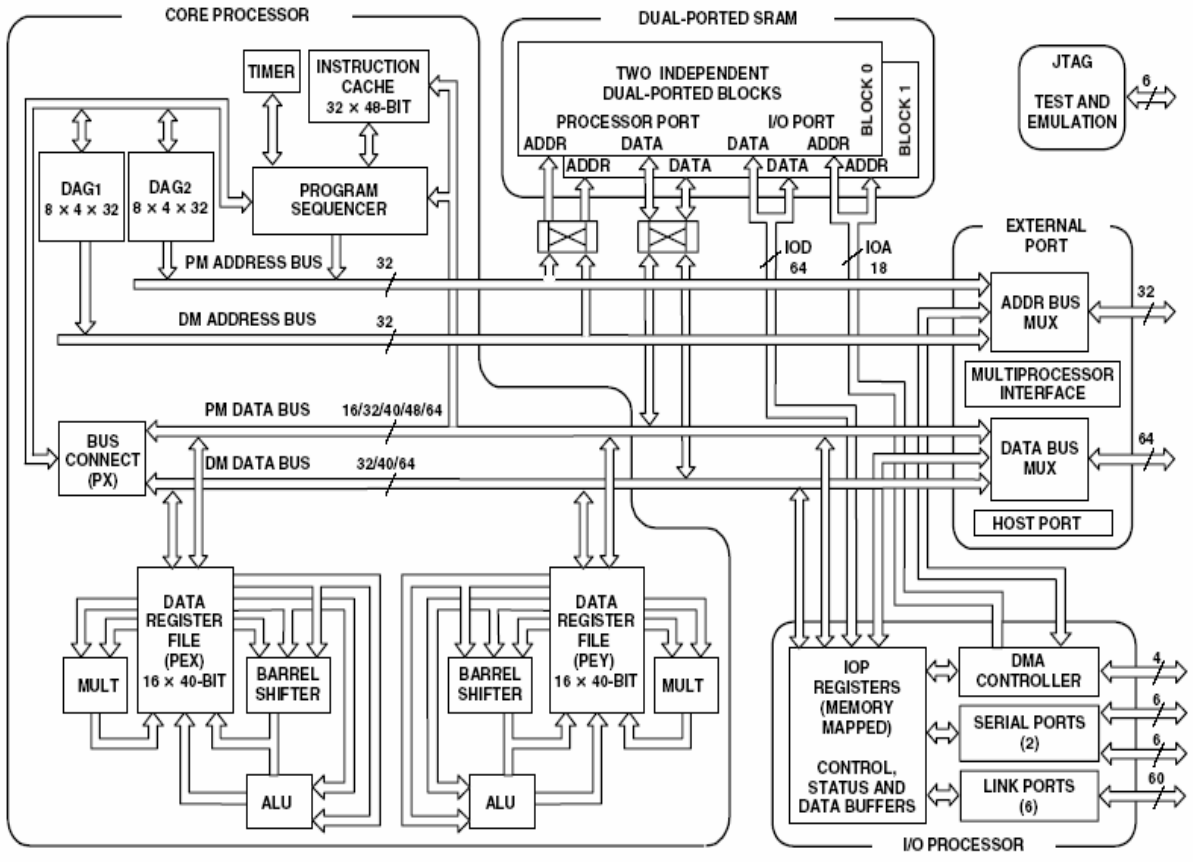


Fig. 5.1 Functional block diagram of ADSP-21160 [15].

The individual modules of the echo synthesis process have been implemented and tested in the simulator, but the interfacing of these individual modules are yet to be done. The following sections present a brief description of the ADSP-21160 architecture followed by the discussions on different software components, used in the echo generation process, in terms of their implementation and computational efficiencies.

5.1 Features of ADSP-21160

The ADSP-21160 is a 32-bit, floating point processor that includes a 100 MHz core, a dual-ported on-chip SRAM, an integrated I/O processor with multiprocessor support, and

multiple internal buses to eliminate I/O bottlenecks. The functional block diagram of ADSP-21160 is shown in Fig. 5.1 and the various important architectural features of the processor are mentioned as below [15].

1) A 10 ns instruction execution time with each instruction is executed in one cycle.
2) On-chip SRAM (4 Mbits) that can be configured in two sets, program memory area and data memory area.
3) Two Data Address Generators (DAG1, DAG2) used for indirect addressing and implementing circular data buffers in hardware. Circular buffers allow efficient programming of data structures required in digital signal processing, and are commonly used in digital filters and Fourier transforms. The two DAGs allow the creation of up to 32 circular buffers with 16 primary registers and 16 secondary. Circular buffers can start and end at any memory location. The address pointer wraparound is automatically handled thus reducing overhead.
4) One interval timer.
5) Two processing elements, referred to as PEX and PEY, each made up of an ALU, Multiplier, Shifter, and Data Register File. This facilitates Single-Instruction, Multiple-Data (SIMD) processing thereby doubling the performance. PEX is always active, and PEY may be enabled by setting the PEYEN mode bit in the MODE1 register. When this mode is enabled, the same instruction is executed in both processing elements, but each processing element operates on different data. This architecture is efficient at executing math-intensive DSP algorithms. Data register file contains 32 registers (16 primary, 16 secondary). The secondary registers are used for fast context switching during interrupt handling. The secondary registers are of the same name as the primary registers. The same register can be used in the interrupt service routine as well as the main program without pushing or popping the contents during the context switching.
6) External port that supports 4 Giga words of 32-bit external memory.

7) DMA controller: 14 zero-overhead DMA channels for background data transfers between ADSP-21160 internal memory and external memory, external peripherals, Host processor, serial ports, or link ports at core clock speed, in parallel with full-speed processor execution.

The ADSP-21160 contains four megabits of on-chip SRAM, organized as two blocks of 2M bits each, which can be configured for different combinations of code and data storage. Each memory block is dual-ported for single-cycle, independent accesses by the core processor and I/O processor. The dual-ported memory in combination with four independent buses allows for dual data fetch, instruction fetch, and non-intrusive, zero-overhead I/O in a single cycle thus making the architecture called Super Harvard Architecture. Moreover, the inclusion of the DMA controllers, ports, and timer, the ADSP-21160 is called a Super Harvard Architecture Computer (SHARC).

On the ADSP-21160, the memory can be configured as a maximum of 128K words of 32-bit data, 256K words of 16-bit data, 85K words of 48-bit instructions (or 40-bit data), or combinations of different word sizes up to four megabits. All of the memory can be accessed as 16-bit, 32-bit, 48-bit, or 64-bit words. A 16-bit floating-point storage format is supported that effectively doubles the amount of data that may be stored on-chip. Conversion between the 32-bit floating-point and 16-bit floating-point formats is done in a single instruction. While each memory block can store combinations of code and data, accesses are most efficient when one block stores data, using the DM bus for transfers, and the other block stores instructions and data, using the PM bus for transfers. Using the DM bus and PM bus in this way, with one dedicated to each memory block, assures single-cycle execution with two data transfers. In this case, the instruction must be available in the cache.

5.2 Software modules: Implementations and Computational efficiencies

This section describes the implementations and computational efficiencies for each software module involved in the echo synthesis approaches described in Sections 4.2.2 and 4.5 in which the Doppler shift is implemented in time domain by segmenting the input data into smaller blocks. All the software modules have been implemented using single-instruction-single-data (SISD) mode of operation of the processor [16].

5.2.1 Data acquisition

The input digital data is acquired from a 12-bit, 2's-complement ADC output by interrupt on IRQ1. The received digital data is first converted to the floating point format before storing in memory. Assuming a 10 V p-p ADC, the received integer data D_i is converted to the floating-point data, D_f, as

$$D_f = (5/2048)D_i$$

The ISR, as shown in Code 5.1, reads each data in 12 instructions (≈ 120 ns).

Code 5.1 ISR to read ADC output

```
DAG register 14 is set as a circular buffer with b14 = start address;

call irq1_intrpt;
nop;
nop;
.
.
.
.segment/pm seg_pmco;
irq1_intrpt:bit clr IMASK IRQ1I;
        nop;
        f4=0.00244140625;           // scale factor for input data = 5.0/2048.0
        f0=dm(0x57ff0);             // ADC address 0x01803002
        f0=f0*f4;                   // multiply data by the scale factor
        pm(i14,m14)=f0;             // store data

        bit set IMASK IRQ1I;
        nop;
        rts;
.endseg;
```

5.2.2 Doppler shift

The evaluation of each sample in the time domain image sequence, implementing 2-point interpolation from the time domain source sequence, needs to execute 15 instructions, or equivalently, takes 150 ns. The sample interpolation program, for L' output points, is given in Code 5.2.

Code 5.2 Sample program for 2-point interpolation

```
.segment/pm seg_pmco;
doppler: f0=alpha;
        r1=L_prime;
        r2=0;
        f3=1.0;
        lcntr=r1, do doppler_shift until lce;
                f5=float r2;                    // m_prime
```

```
                f5=f5*f0;                       // f5=m
                r15=trunc f5;                   // r15=m0
                m15=r15;
                f6=float r15;
                f5=f5-f6;                       // m-m0
                f7=f3-f5;                       // 1-(m-m0)
                i13=i12;
                modify(i13,m15);
                f8=pm(i13,m13);                 // inp(m0)
                f10=f7*f8, f9=pm(i13,0);        // inp(m0+1)
                f11=f5*f9;
                f12=f10+f11;
                dm(i0,m0)=f12;
                doppler_shift: r2=r2+1;
                rts;
.endseg;
```

5.2.3 FFT

A radix-2 DIT FFT has been implemented with the built-in hardware facility of the processor for implementing bit-reversal thus requiring no extra software overhead. The bit-reversal algorithm, for storing N data from a sequential memory block to a memory block with bit-reversed address, is implemented as given in Code 5.3.

Code 5.3 Bit-reversal module

```
#define bitrev_modifier  0x00100000;        // 4k -> 12, hence 12 th bit from left
.
.
.
r0=N;
r1=bitrev_modifier;
b12=input; m12=1;l12=0;                     // to read the stored data
i0=inp_real; m0=r1; l0=0;
lcntr=r0, do bit_reversal_real until lce;
        f2=pm(i12,m12);
        bit_reversal_real:dm(i0,m0)=f2;
```

The computation times for the radix-2 DIT FFT algorithm, for different length of input data, N, are shown in Table 5.1.

5.2.4 Addition or multiplication operation

While implementing the addition operation, $c_i = a_i + b_i$ for $i = 0, 1, 2, \ldots$, one addition operation involves different computational efficiencies depending upon where, in data or program memory, the operands and results are intended to be stored. The efficient

technique involves storing the operands in DM (or PM) and the results in the other block of memory, i.e., PM (or DM). Thus, each addition operation requires 2 instructions (≈ 20 ns). The same holds true for multiplication operation. The sample program, for i = 0, 1, …, N-1, is given in Code 5.4.

Code 5.4 Point-by-point addition of two blocks of data

```
f0=dm( );
f1=dm( );
lcntr=N, do add_loop until lce;
        f4=f0+f1, f8=dm( );
        f9=dm( );
        f12=f8+f9, pm( )=f4, f0=dm( );
        add_loop: pm( )=f12, f1=dm( );
```

Table 5.1 Computation times for radix-2 DIT FFT algorithm

N	256	512	1k	2k	4k	8k	16k
Time (ms)	0.13	0.32	0.72	1.7	3.4	7.12	14.4

5.2.5 MAC operation

In Section 5.2.4, it is seen that each addition or multiplication operation requires 2 instructions. However, since ADSP-21160 has a parallel multiplier and accumulator blocks, each MAC operation, $d_i = a_i + b_i * c_i$ for i = 0, 1, …, can be performed in 2 instructions provided that c_i and d_i are stored in the same memory block, that is, either in PM or DM.

5.2.6 SNR enhancement

Assuming $X = X_{re} + jX_{im}$ and $Y = Y_{re} + jY_{im}$ be the input and output complex numbers, respectively, of the SNR enhancement block, the generation of each output involves two square operations and a summation operation to evaluate $|X|^2$, evaluation of the phase, $\theta = \tan^{-1}(X_{im}/X_{re})$, one subtraction operation, one square-root operation, evaluation of $\sin\theta$ and $\cos\theta$, and finally two multiplication operations. Taylor's method, of infinite series expansion of a function, is used to evaluate θ, $\sin\theta$ and $\cos\theta$. The real and imaginary

values, thus obtained, are multiplied by the sign of X_{re} to obtain Y. The complete operation of generating each complex output requires 47 instructions (≈ 470 ns).

5.2.7 Echo extension using the method described in Section 4.2.2

In this method, let it be assumed that the input data and target highlights are segmented into blocks of equal lengths, L_p, and the FFT outputs of each block of the target highlights are pre-calculated and stored in memory. The echo extension then consists of the complex multiplications, N_p-point IFFT operations, and overlap-add operations as shown in Table 4.2. Each complex multiplication involves four real multiplications and two real additions thus requiring 8 instructions (≈ 80 ns). Since IFFT operation is same as FFT operation, the computation times for the echo extension operations, for different values of N_p, are given in Table 5.2.

Table 5.2 Computation times for echo extension operation

N_p	L_p	Number of Blocks, $B=M/L_p$	$B(N_p/2+1)$ comp. mult. (ms)	B times N_p–point IFFTs (ms)	Overlap-add (ms)
16 k	8 k	3	1.96	43.2	1.8
8 k	4 k	6	1.96	42.7	0.9
4 k	2 k	12	1.96	40.8	0.45
2 k	1 k	24	1.96	40.8	0.23

5.2.8 Echo extension using the method described in Section 4.5

As shown in Table 4.6, the echo extension operation consists of L_p-point IFFT and overlap-add operations. The computation times for the echo extension operations, for different values of L_p and n_d, are shown in Table 5.3.

5.3 Discussion

The complete echo synthesis process can be implemented in real-time if the time to generate the echo, corresponding to a block of input data, is not more than that required to receive L_s input points.

The computation time during the process of echo extension, in frequency domain by segmenting both the input data and target highlights, does not change considerably by either increasing or decreasing the number of blocks in target highlights. This is because of the fact that the computation times for the IFFT operations due to segmentation of the target highlights remains same irrespective of the total number of blocks. By reducing the number of blocks in the target highlights, that is, by increasing the size of the blocks of the input data and target highlights, the computation time for echo extension reduces compared to the time to receive the input samples. But, the time to process these input data prior to echo extension increases. Moreover, the delay in outputting data increases due to the initial collection time of long data sequence. The real-time implementation of echo synthesis, using overlap-add method and Doppler shifting performed in time domain with n_d = 1, is possible if L_p = 17 k. Hence, the real-time implementation of the echo synthesis algorithms, with echo extension implemented in frequency domain, is difficult in the proposed hardware based on ADSP-21160.

Table 5.3 Computation times for echo extension operation

L_p	L_p-point IFFT (ms)	Overlap-add	
		n_d = 10 (ms)	n_d = 20 (ms)
8 k	7.12	394	196.8
4 k	3.4	197	98.4
2 k	1.7	98.5	49.2
1 k	0.72	49.25	24.6

The time-domain approach of implementing the echo extension, for n_d > 1, involves the computation time that is directly related to L_p and inversely related to n_d, as shown in Step 5 of Table 4.6. Since the processing time prior to the echo extension operation also reduces with L_p, the total computation time for the echo synthesis can be reduced by decreasing L_p and increasing n_d. However, reducing L_p calls for a reduction in

time to collect the input data and hence, the real-time implementation of the echo synthesis process can be achieved by suitably increasing n_d. The real-time implementation of this approach of echo synthesis is possible if the factor $(M\text{-}1)/n_d + 1$ is constant thus making the entire operation dependent only on input data block and independent of the length of the target highlights. For example, if L_p = 1k, i.e., L_s = 1003, f_s = 200000 k sa/s, M = 24000, and n_d = 130, the total echo synthesis operation takes 5 ms and the sequence of L_s input points also arrives in 5 ms.

Chapter 6

SUMMARY AND CONCLUSION

In this report, an algorithm is proposed for an active sonar application such that, if implemented, a stationary and small electronic object, called decoy, can be made to appear as a moving and large object to a torpedo operating in active mode. The decoy detects the incoming signal, enhances it, introduces the required Doppler shift and echo extension in the enhanced signal corresponding to a suitable velocity and size of a target, and finally transmits the simulated echo signal. All these operations have been implemented without analyzing the signal that is received by the decoy. The signal enhancement is achieved through spectral noise subtraction method. The operation of imparting Doppler shift to a signal has been attempted in time domain by a simple 2-point interpolation method and also in frequency domain by adopting two different methods, frequency sample and segment mapping methods. It is seen that a large amount of computation as well as storage are required if Doppler shift is implemented in frequency domain. The extension of the echo from the target is incorporated by introducing different delays in the target highlights. It is seen that the echo extension operation can not be implemented efficiently even in frequency domain if the length of the target highlights is too large. If the length of the target highlights is too large, however, the time domain implementation appears effective only if the delay introduced in the target highlights is also more. Three schemes, based on different combinations of implementing the Doppler shift and echo extension, have been mentioned in Chapter 4 for echo simulation with each scheme needing different amount of computational complexities and storage. The complete algorithm has been implemented using MATLAB, and tested with signals such as PCW, LFM, and CFS in additive white Gaussian noise.

Finally, the individual modules of the entire algorithm have been implemented, through software, for a real-time application in a DSP hardware based on ADSP-21160

digital signal processor. The real-time software has been developed in assembly language and the computation time for each software module has been calculated with respect to the instruction execution time of the processor and the number of instructions performed by the module. The real-time implementation of the decoy software has been done by considering that the input data is segmented into smaller blocks and the Doppler shift is implemented on each of these blocks in time domain. However, the echo extension operation has been implemented in frequency domain by segmenting the target highlights as well as in time domain by considering the entire length of the target highlights with equal delay between adjacent non-zero values.

It is concluded that the operations of Doppler shifting and signal enhancement can be implemented, in real-time, on the proposed hardware based on ADSP-21160 processor, by segmenting the input data sequence in smaller blocks and performing Doppler shifting on each block of input data in time domain. The approach also needs less memory in the hardware. But, it is difficult to implement the echo extension operation in real-time on this hardware either in frequency domain or time domain and whether the target highlights are segmented into smaller blocks or not if the target highlight has a large length and $n_d = 1$. However, for $M/n_d \approx 184$, the hardware just performs the complete echo synthesis operations in real time for any length of target highlights if the echo extension is implemented, with the entire block of the target highlights, in time domain.

REFERENCES

[1] *Underwater Signal Detection – 2002, Internal document*, Naval Science and Technological Laboratory, Visakhapatnam, Ministry of Defence, Govt. of India.

[2] S. A. Tretter, "Estimating the frequency of a noisy sinusoid by linear prediction," *IEEE Trans. Inform. Theory,* vol. IT-31, pp. 832-835, Nov. 1985.

[3] Petar M. Djuric and Steven M. Kay, "Parameter estimation on Chirp signals," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 38, no. 12, Dec. 1990.

[4] G. S. Gill and J. Huang, "The ambiguity function of the step frequency radar signal processor," *Proc. of CIE International Conference of Radar,* 1996.

[5] T. H. Glisson and Charles L. Black, "On digital replica correlation algorithms with applications to active sonar," *IEEE Trans. on Audio and Electroacoustics*, vol. AU-17, no. 3, Sep. 1969.

[6] R. J. Urick, *Principles of Underwater Sound*, New York: McGraw-Hill, 1975.

[7] M. I. Skolnik, *Introduction to Radar System*, New Delhi: Tata McGraw-Hill, 4th ed, 1998.

[8] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications,* New Delhi: Prentice Hall of India, 3rd ed, 1996.

[9] S. Boll, "Suppression of Acoustic noise in Speech using Spectral Subtraction," *IEEE Trans. On Acoustic, Speech and Signal processing*, vol. 27, pp. 113-120, Apr. 1979.

[10] M. Berouti, R. Schwartz and J. Makhoul, "*Enhancement of Speech Corrupted by Acoustic Noise*," Proc ICASSP, pp. 208-211, 1979.

[11] Roger W. Schwenke, “Sensitivity analysis of an Estimator-correlator for the detection of spread targets with multiple discrete highlights,” Ph.D. thesis, Dept. of Acoustics, The Pennsylvania State University, Dec. 2000.

[12] Boo Kim, H. Uk Lee and M. H. Park, “A study on highlight distribution for Underwater Simulated Target,” *IEEE International Symposium on Industrial Electronics*, vol. 3, pp. 1988-1992, Jun. 2001.

[13] Shi Lin Pan, Qiang Wang, Zay Q. Li and Yi H. Zhai, “The echo data simulation of airborne DBS radar,” *Sixth International Symposium on Antennas, Propagation and EM theory*, pp. 608-611, Nov. 2003.

[14] W. Cui, He Chen and Y. Han, “VLSI implementation of universal random number generator,” *Asia-Pacific Conference on Circuits and Systems*, vol. 1, pp. 465-470, Oct. 2002.

[15] *ADSP-21160 SHARC processor hardware reference*, Analog Devices Inc., USA.

[16] *ADSP-21160 SHARC processor instruction set manual*, Analog Devices Inc., USA.