Architectures for Real-Time On-Board Synthetic Aperture Radar Processing

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. ir. K.F. Wakker,
in het openbaar te verdedigen ten overstaan van een commissie,
door het College van Dekanen aangewezen,
op maandag 4 december 1995 te 13.30 uur

door

Laurens Herman Jozef BIERENS

elektrotechnisch ingenieur
geboren te Amsterdam

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. ir. P.M. Dewilde

Toegevoegd promotor:
Dr. ir. E.F.A. Deprettere

Samenstelling promotiecommissie:

Rector Magnificus, voorzitter
Prof. dr. ir. P.M. Dewilde, TU Delft, promotor
Dr. ir. E.F.A. Deprettere, TU Delft, toegevoegd promotor
Prof. dr. ir. J. Biemond, TU Delft
Prof. dr. ir. L.P. Ligthart, TU Delft
Prof. dr. ir. H. Blok, TU Delft
Dr. A. Farina, Alenia, Italië
Ir. J.P. van Bezouwen, TNO-FEL

Dit proefschrift is mede tot stand gekomen dankzij de ondersteuning van TNO.

CIP-DATA KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Bierens, Laurens Herman Jozef

Architectures for real-time on-board synthetic aperture
radar processing / Laurens Herman Jozef Bierens. - Den
Haag : TNO Fysisch en Elektronisch Laboratorium. - Ill.
Thesis Technische Universiteit Delft. - With ref.
ISBN 90-6743-402-7
NUGI 832 Subject headings: synthetic aperure radar / remote
sensing / real-time processing

Copyright ©1995 by Laurens Bierens.
All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, elctronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission in writing from the author.

Stellingen behorende bij het proefschrift

Architectures for Real-Time On-Board Synthetic Aperture Radar Processing

door

Laurens Bierens

1. If DSP hardware designers would use "processing speed per units of processor volume and power consumption" as a measure for the processing performance of their hardware instead of "operations per seconds", then the popularity of DSP boards would decrease dramatically.

2. A designer does not serve the application of an algorithm if he writes out multi-indexed formulas and equations in full low-level detail. only showing his ability in performing difficult actions, but he serves it well if he uses graph manipulations, which allows him to concentrate on typical application specific constraints.

3. Rapid prototyping of a VLSI implementation using DSP boards is only profitable if a hardware designer has extensive experience with these DSP boards and their development tools. Otherwise realization of the VLSI implementation itself is the cheapest way of rapid prototyping.

4. It does not make sense to specify a radar signal processing system in full detail, if the specifications of the radar front-end keep changing.

5. The term "pulse compression" is an inheritance from the days that knowledge about digital signal reconstruction techniques was limited. Nowadays it merely narrows the scope of radar system engineers.

6. If electrical engineering would be propagated more as the basics of what secondary school pupils concerns (Nintendo, MTV, Internet, house music), then more of them would choose to study it.

7. Just as the sensitivity of the inversion of a matrix can be disturbed by its small singular values, so can the sensitivity of a large social inversion process be disturbed by insignificant individuals in society. A poignant example is the murder on the Israeli prime-minister Yitzhak Rabin committed by the law student Yigal Amir.

8. The fact that fashion reverts to the past, implies that being behind the fashion will always give a lead.

9. If during rush-hour on stations the train compartments would have a direct connection with the platforms (as is often the case with subways). instead of with narrow and crowded gangways, then the delays of the trains would be reduced dramatically.

10. Due to the expected increase in the use of Windows 95. memory chips are currently reliable objects for investment.

to Annemieke

CONTENTS

1 INTRODUCTION **7**

1.1 Objective and Motivation 7

1.2 Outline of this Thesis 10

2 INTRODUCTION TO SAR PROCESSING **12**

2.1 Introduction 12

2.2 Doppler Analysis 14

2.3 Pulse Compression 18

2.4 Azimuth Compression 19

3 SAR PROCESSOR SPECIFICATIONS **28**

3.1 Introduction 28

3.2 SAR Processing System Parameters 30

3.3 Real-Time SAR Processing for PHARUS 35

3.4 Real-Time SAR Processing for ERS-1 39

3.5 Concluding Remarks 42

4 CONVOLUTION AND MULTIRATE SIGNAL PROCESSING **44**

4.1 Introduction 44

4.2 Notations and Preliminaries 45

4.3 Multirate Convolution Systems 48

4.4 Multirate Convolution and the DFT 55

4.5 Concluding Remarks 60

5 CONVOLUTION AND BLOCK-SIGNAL PROCESSING **62**
5.1 Introduction 62
5.2 The One-Dimensional Convolution Problem 63
5.3 Sectioning of Large Convolution Problems 65
5.4 Short Length FFT Processing 73
5.5 Concluding Remarks 80

6 HARDWARE ARCHITECTURE DESIGN **81**
6.1 Introduction 81
6.2 Algorithm Specification 82
6.3 Prototype Architecture 86
6.4 Dedicated VLSI Architecture 92
6.5 Concluding Remarks 101

7 CONCLUDING REMARKS **102**

A A MATRIX ALGEBRAIC VIEWPOINT OF SAR **106**

B DECONVOLUTION SYSTEMS **114**

C MOTION COMPENSATION **124**

D OVERLAP-ADD/DISCARD CONVOLUTION NETWORKS **133**

BIBLIOGRAPHY **137**

SAMENVATTING **144**

CURRICULUM VITAE **146**

CHAPTER 1

INTRODUCTION

1.1 Objective and Motivation

This thesis is about the design of architectures for *real-time on-board Synthetic Aperture Radar (SAR)* processing. A SAR is a side-looking imaging radar on a moving platform, in general an aircraft or satellite. The SAR principle is based on synthetically generating an effective long antenna by signal processing rather than using a long physical antenna. The actual physical antenna length is in most cases relatively small. Generating a SAR image requires two-dimensional signal processing techniques. The model of the SAR system that we use consists of a *SAR acquisition system* and a *SAR reconstruction system*. The acquisition system maps the information of the Earths surface onto a 2D SAR echo signal. The reconstruction system maps the 2D SAR echo signal onto a SAR image. *SAR processing* or *SAR imaging* usually refers to the SAR reconstruction system.

Real-time on-board SAR processing

The work presented in this thesis is strongly related to the PHARUS system. PHARUS stands for Phased Array Universal SAR and is a fully polarimetric C-band airborne SAR with an active antenna array [HSKP92]. The PHARUS system uses a phased array antenna, which provides in a flexible design and a compact and light-weight antenna system. The latter allows the user to mount the PHARUS system on a small aircraft. The use of a small aircraft will improve the operational cost/benefit ratio of an airborne SAR. Typical applications of the PHARUS system range from agricultural classification and geophysical mapping to typical defense applications as surveillance, reconnaissance and Fixed/Moving Target Indication (FTI/MTI). Moreover, there is the perspective of using PHARUS as a demonstrator for ESA's future Advanced SAR (ASAR) system, a polarimetric spaceborne SAR system. At the moment of writing the PHARUS system has concluded its first test-flight successfully.

The motivation for the work presented in this thesis was the need for a real-time on-board SAR processor for the PHARUS system. The specifications of the real-time on-board SAR processor depend on the application. We give a representative, albeit incomplete, listing of applications of the PHARUS system for which a real-time on-board SAR processor is essential.

Surveillance, reconnaissance: Surveillance and reconnaissance are typical defense applications, but it may also be important for e.g. coast guarding and pollution monitoring. The main requirement is the fast detection of suspected objects or events. The processing time is thus critical, in that it must be real-time, and processing must be performed on-board the platform. If the platform is small, e.g. a fighter or an Unmanned Airborne Vehicle (UAV), then low power dissipation and small processor volume are additional constraints.

SAR data transmission: On-board processing is mandatory, due to the extremely broadband communication links that are needed for the transfer of the uncompressed radar data to the Command & Control Centers below. In defense application, the transferred data may be (partly) corrupted due to *jamming*. On-board processing of the SAR data (in combination with image processing) is a rigorous form of data compression. It allows one to send only small banded signals to the Command & Control Centers. These small band communication links are, in potential, less sensistive to jamming.

FTI/MTI: FTI/MTI require high geometric resolution SAR images, often combined with real-time target detection ability. Moreover, for MTI purposes also the phase of the image is needed to obtain information about the movement of the detected objects. Roughly, the same constraints hold for FTI/MTI as in surveillance and reconnaissance, with the addition that processing capacity is more critical. This is due to the higher geometrical resolution (for both FTI and MTI purposes) and the higher pulse repetition frequency (for MTI purposes) that are required.

On-line system monitoring: Despite the application specific requirements, it is generally preferable that a SAR system includes an on-board real-time SAR processor, so that on-line monitoring is possible. A SAR system might malfunction due to technical defects, or the surface area illuminated by the antenna might be wrongly chosen. Since SAR data acquisition requires expensive missions, an early discovery of malfunctioning or a wrong flight trajectory can save the operator a lot of expenses.

From the list we can determine some preliminary specifications for a real-time on-board SAR processor for the PHARUS system. First of all, the processor must meet the specified processing speed in all cases. During the movement of the platform, the Earths surface is being illuminated by the SAR antenna. Obviously, we have some delay time between illuminating a point of the

Earths surface and displaying the SAR image pixel corresponding to this point. We say that SAR processing is performed in *real-time* if this delay time is not more than $\mathcal{O}(10)$ seconds, and is independent of the size of the illuminated area.

The second specification is that the SAR processor size must be small and have limited power consumption. High-performance real-time on-board SAR processors do already exists, see e.g. [DJCM92, Mer95]. However, their size and power consumption is substantial, in that they will not fit in small platforms such as fighters and UAVs. Our intention is to develop hardware for real-time on-board SAR processing that potentially fit in these small platforms. We even opt for a real-time on-board satellite SAR processor in the near future.

A third specification is that the SAR reconstruction system is linear. Reduction of computational complexity in SAR processing is often obtained by simplification of the SAR acquisition system model, which leads to non-linear SAR reconstruction systems, see e.g. [CPR89, Mor92]. These simplifications may lead to non-linearities in the SAR image. An example is the occurance of side-lobes at (sometimes unpredictable) positions far from the main-lobe of a point target response. These are assumed to be negligible for several imaging purposes. However, for MTI purposes it is essential to have an accurate model of the SAR reconstruction system. Moreover, it is desirable to have a linear system model of the reconstruction system, so that the analysis of the phase history of point targets is straight-forward. For this reason, we consider the model of the SAR acquisition and reconstruction system presented in [WLJ82] as optimal, since it describes the impulse response in terms of linear operations, namely 2D convolutions.

Approach of the work

Our approach in designing a real-time on-board SAR processor is to emphasize on small and high speed hardware. Low power consumption is considered as an additional constraint. The specifications of the SAR acquisition system are given, however, we do not address the IF and RF system configuration, the atmospheric influences, nor the antenna system specifications. The SAR processing specifications are based on the PHARUS antenna system configuration, although, we do allow for some flexibility. They also depend on the operation modes of PHARUS, but operation modes may change. Therefore we will use the specifications as a *guidance*: they determine the *design approach* of the real-time on-board SAR processor , but they do not prescribe the *absolute* specifications of the real-time SAR processor.

Consequently, this thesis will not end up with a design of a real-time on-board SAR processor. We determine the critical processing steps required for real-time on-board SAR processing, and propose an effective VLSI-based solution for it. The critical processing step within this real-time on-board SAR processor appears to be 1D correlation/convolution of long discrete signals with a high effective data rate. The solution that we propose, reduces the size of the required hardware for the critical processing step from several DSP boards to a few chips. Obviously, power consumption also reduces proportional to this reduction. The design approach is generic, and may even bring real-time on-board satellite SAR processing within reach.

1.2 Outline of this Thesis

We give a brief overview of this thesis per chapter. The objectives and the motivation of the work have been presented in this chapter.

In chapter 2 we briefly review the basics of SAR processing. The impulse responses of the SAR acquisition and reconstruction systems that we derive are based on the model presented in [WLJ82]. The SAR basics are addressed in terms of Doppler bandwidth and resolution.

In chapter 3 we determine the SAR processor parameters. We give the relations between the SAR processor parameters on the one hand and the SAR system specifications and the required image resolution on the other hand. We present two illustrative examples: a real-time on-board airborne SAR processor for the PHARUS system and for real-time on-board spaceborne SAR processor for the ERS-1 system.[1] We analyze the SAR processing data flow with respect to the SAR system specifications and the required SAR image specifications. Based on this analysis, we show how and where data reduction within the data flow can be achieved, without affecting the SAR image specifications. We conclude this chapter with the notification of the critical processing step within the SAR processing: the 1D correlation/convolution.

The 1D convolution problem is addressed in chapter 4. We introduce the multirate convolution system to solve the long convolution problem. It is known that multirate filter banks have a close relationship with short-time Fourier transforms and fast convolution schemes [Vet88]. We elaborate this relationship in terms of combined algorithm development and architecture design. The result is a design methodology that allows direct mapping of an algebraic specification of signal processing algorithms - here multirate convolution - onto prototyping architectures or dedicated VLSI processors. This mapping is obtained by transformations of graphical representations rather than by manipulating multi-indexed formulas.

In chapter 5 we show the exact relationship between the multirate convolution system and block-signal processing algorithms. The latter are traditionally used to solve the long convolution problem, the so-called overlap-add and overlap-discard methods. Here we exploit the graphical representation to map the long convolution onto a highly regular architecture based on short FFTs. This is in fact a schematic description of the multirate convolution system.

In chapter 6 we implement the multirate convolution system in an efficient VLSI hardware architecture. The first stage in the design trajectory is the design of a prototype architecture of the convolution processor based on off-the-shelf DSP components. This stage is essential to gain hands-on experience in the mapping of the system onto hardware. Based on the prototype architecture a single-chip convolution processor is designed. The chip convolves up to a maximum of 8 Ksamples convolution length with an effective data rate of 2 Msamples/sec. This performance compares roughly to an off-the-shelf DSP board. However, it is obvious that the size and power consumtion of a DSP board is much more than the size and power consumtion of a single chip. Moreover, the convolution data rate of our chip can be increased by parallel

[1]ERS-1 is a satellite Remote Sensor of the European Space Agency (ESA).

processing. The used arithmetic is based on a hybrid floating point data format.

In chapter 7, we round off this thesis. We briefly address how the main results of this thesis are implemented. Also we give some hints for solving some open problems: motion compensation and autofocusing.

CHAPTER 2

INTRODUCTION TO SAR PROCESSING

2.1 Introduction

The performance of the SAR processing strongly depends on the parameters of the SAR antenna system, such as wavelength, antenna size, etc.. But the platform parameters are as important as the antenna system parameters. Examples of platform parameters are speed and altitude as well as unpredictable platform motions. All have their impact on the signal processing effectiveness, and that is what this chapter is about: the description of SAR in terms of system parameters, such that we can determine processing strategies.

In this chapter we will concentrate on purely predictable SAR system parameters and their impact on the processing of SAR data. In fact, these parameters are basic to the SAR principle. The movement of the SAR platform with a certain constant speed introduces a Doppler effect. This Doppler effect is implicitly exploited in enhancing the resolution of a SAR image in the direction of the movement. This principle was first mentioned by Sherwin in 1952 [SR62], and has been validated in later years by e.g. Cutrona et al [CVLH61].

However, the system parameters introduce also some unwanted side-effects, such as *range migration* and *range-dependent attenuation.* The first effect is caused by the fact that the echo signal of a point on the ground, is shift-variant in range direction. Only in case of low-quality SAR imaging this effect might be negligible. Otherwise we have to introduce 2D processing techniques to compensate the range migration effect. The second effect is a result of the free space attenuation of the electromagnetic waves. The analysis of the attenuation in relation with transmitted power and system noise gives the boundary condition of the maximum range of the SAR system.

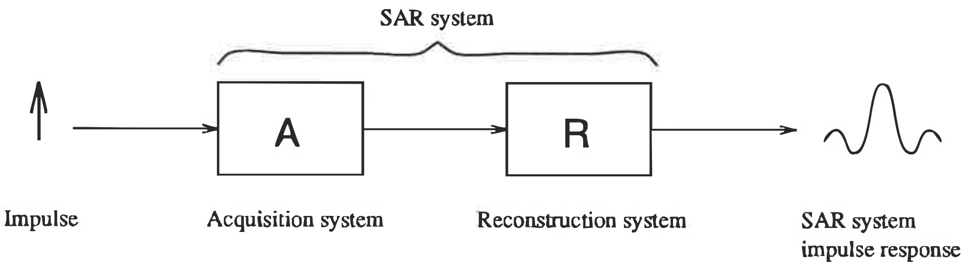


Figure 2.1: The modelling of the impulse response of the SAR system. A denotes the impulse responses of the SAR acquisition system, which includes the transmitter, the atmosphere, the receiver, and the A/D conversion. R denotes the impulse response of the reconstruction system: the SAR imaging system.

The SAR system specifications can be determined from the so-called SAR system impulse response. To determine the SAR system impulse response, it is generally assumed that the imaged surface contains a single point target, which is illuminated by the antenna beam. The mathematical equations required to reconstruct the point target can be obtained from the analysis of the radar returns from this point target. The complete system is shown in figure 2.1, where A stands for the impulse reponse of the data acquisition system and R stands for the impulse response of the reconstruction system, also called the SAR imaging system.

The objective is now clear: find a mathematical description of the SAR acquisition system A and determine from this the reconstruction system R. Of course, in the ideal world R would be the inverse function of A. However, in the real world this inversion is hardly possible due to system limitations such as the limited bandwidths in both range and azimuth direction and the appearance of thermal noise within the echo signal.

The viewpoint we take in this chapter is the classical Doppler viewpoint, which is well-understood and well-described in the literature (excellent references are e.g. [Cut70, Oli89, CM91]). We do not intend to introduce any novel viewpoint on the SAR principle, we shall however strive for a novel viewpoint on development of fast and small hardware architectures for the SAR processing. Nevertheless, in appendix A we will give a brief description of SAR from a matrix algebraic viewpoint, see [BD92].

The line of thought of this chapter is along the step-wise analysis of SAR data, leading to a parametric description of the impulse response of the ideal SAR system. The Doppler viewpoint is reviewed in section 2.2, where also some fundamental conditions of the SAR parameters are examined. In section 2.3 the signal processing technique known as *pulse compression* is reviewed. It achieves enhancement of the resolution in the radar look-direction. In section 2.4 it is shown how the Doppler effect can be exploited to enhance the resolution in the flight direction. Including the range migration in this description results in a model of the SAR system impulse that is presented in [WLJ82].

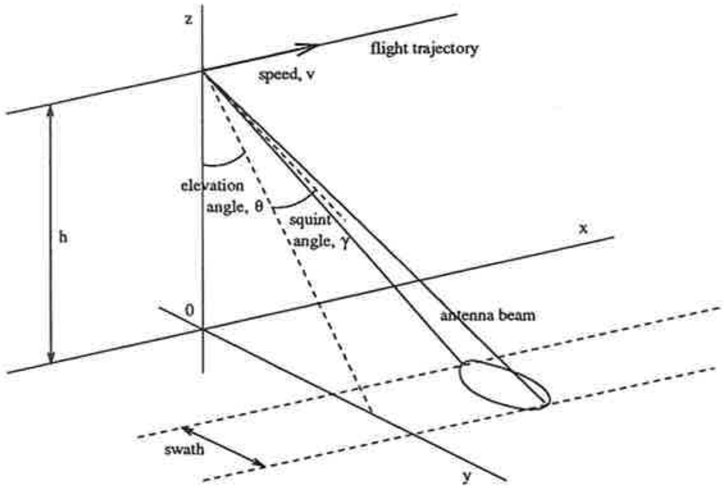


Figure 2.2: Geometry of the SAR system.

2.2 Doppler Analysis

In this section we take a closer look at the fundamentals of synthetic aperture radar. We analyze the radar returns in terms of Doppler shifts. This will lead to the high-resolution imaging property of SAR. In the analysis we assume perfect conditions, i.e. we describe the steps that are required in SAR processing mathematically, thereby neglecting the inaccuracies due to range-dependent attenuation, range migration and unwanted platform motions.

Geometry

The geometry of the SAR system that we will consider is shown in figure 2.2. The moving platform will be an aircraft. Let points in 3D space be specified in Cartesian coordinates (x, y, z). The earths surface is represented by the xy-plane, and the antenna platform is moving with speed v along the straight flight trajectory $\{(x, y, z)|y = 0, z = h\}$, where h is the altitude of the platform above the earths surface. The relation between the platform's x-coordinate and its speed is given by $x = vt$, where t is the time variable. The elevation of the antenna beam is measured by the angular inclination θ, relative to the z-axis. The squint of the SAR antenna is the deviation of the radar look-direction from the yz-plane. It is measured by the squint angle γ, see figure 2.2. During the flight the SAR continuously transmits pulses and receives their echoes. The points on the ground that are illuminated by the successive pulses during the flight are referred to as ground swath. In SAR processing it is customary to use the terms range and

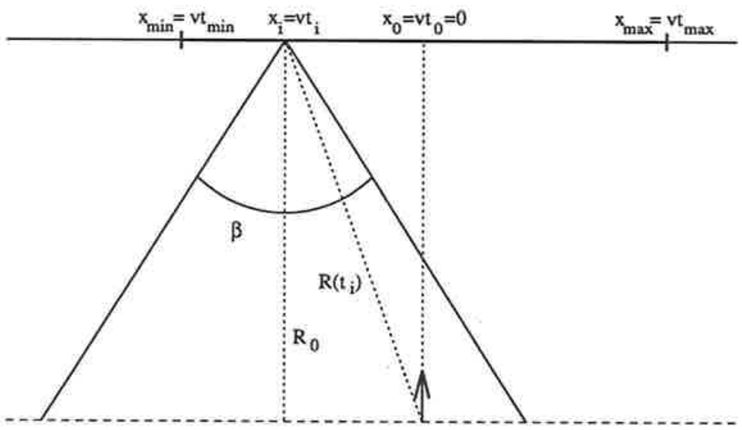


Figure 2.3: The simplified geometry in azimuth of the SAR system. β is the beamwidth of the physical antenna in azimuth. The point target is within the antenna beam as long as the x-coordinate of the antenna position is within the interval (x_{min}, x_{max}).

azimuth. Range is defined as the distance between the antenna and a point on the xy-plane. Azimuth is defined as the direction along the line of flight of the aircraft.

The SAR signal space is usually taken to be temporal and two-dimensional, that is, it is expressed by the 2D time variable (t', t). The variable t' is related to the round-trip of the transmitted signal and is roughly confined to time-intervals of $\mathcal{O}(10^{-4})$ seconds. The variable t is related to the movement of the platform and is confined to time intervals in the order of seconds or more. They are linearly related to the two spatial variables of the so-called object space: the terrain that is to be imaged. The 2D object space is characterized by the range and azimuth variable (R, x). In the following we derive the relations between the variables.

The 2D echo signal

Let $p(t)$ be a single pulse, non-zero in the time interval $(-\tau_p/2, \tau_p/2)$ and zero otherwise, where τ_p is the pulse width. The pulse is transmitted in range direction. Let f_{prf} be the *pulse repetition frequency*, then the time interval between two subsequent transmitted pulses, also called the *inter-pulse time*, is $T_{prf} = 1/f_{prf}$. Let t_i, $i = \cdots, -1, 0, 1, \cdots$, be the time instants at which the pulses are transmitted, using a carrier frequency ω_c. For each i, the transmitted signal is then

$$s_i(t) \quad = \quad p(t - t_i)\exp(j\omega_c t) \qquad (2.1)$$

It is convenient for the Doppler analysis to assume that $\gamma = 0$, see figure 2.3. The case $\gamma \neq 0$ will be discussed in section 2.4. Furthermore, let $x_i = vt_i$ be the x-coordinate of the antenna position at t_i and assume, for convenience, that $t_0 = 0$ and thus also $x_0 = 0$. Consider a point target on the ground with x-coordinate zero. The distance between the antenna position and the point target is function of the time, and denoted as $R(t)$. The *nominal range* R_0 is defined as the distance between the antenna and the point target at t_0, and thus in this case $R_0 = R(0)$.

Let D be the length of the physical antenna in azimuth and let $\lambda = 2\pi c/\omega_c$ be the wavelength of the carrier wave, then the physical half-power beamwidth in azimuth is $\beta = \lambda/D$ [Cut70]. In practice, β is small, thus the *spatial* beamwidth of the antenna beam in azimuth at distance R_0, say L_{illum}, approximates

$$L_{illum} = \beta R_0 = R_0\lambda/D \tag{2.2}$$

Let $x_{min/max} \equiv vt_{min/max} = \mp \frac{1}{2}L_{illum} = \mp R_0\lambda/2D$, see figure 2.3, then the pulse transmitted at t_i is reflected back to the antenna if $x_{min} \leq x_i \leq x_{max}$. We consider all i for which the latter inequality holds.

If the atmospheric attenuation is assumed zero, then, for each i, the echo signal is a shifted version of $s_i(t)$, where the shift is determined by the round-trip delay of the transmitted signal. Let c be the speed of light, then the round-trip delay of the pulse transmitted at t_i is $2R(t_i)/c$. For each i, the echo signal received by the antenna is then

$$s_i(t - 2R(t_i)/c) \simeq p(t - t_i - 2R(t_i)/c)\exp(-j\omega_c 2R(t_i)/c) \tag{2.3}$$

In the latter expression we have omitted the term $\exp(j\omega_c t)$, since it bears no additional information of the point target and is removed by demodulation. Thus, the 1D echo signal of the subsequent pulses is the summation

$$\sum_i s_i(t - 2R(t_i)/c) \simeq \sum_i p(t - t_i - 2R(t_i)/c)\exp(-j\omega_c 2R(t_i)/c) \tag{2.4}$$

However, since SAR is an imaging radar, we rather work with a 2D echo signal. If we use the substitution $t - t_i \leftarrow t'$, for all i, with $t' < T_{prf}$, then we can write the 2D echo signal as

$$e(t', t_i) \triangleq s_i(t' + t_i - 2R(t_i)/c) \simeq p(t' - 2R(t_i)/c)\exp(-j\omega_c 2R(t_i)/c) \tag{2.5}$$

In practice, this means that for each transmitted pulse, the pulse echo is acquired until the next pulse is transmitted. The acquisition time per pulse echo is thus limited by the inter-pulse time.

Doppler frequency

If the distance between the signal source and the signal receiver changes in time, then it induces a frequency shift of the received signal, known as the *Doppler shift* or *Doppler frequency*. This Doppler frequency is proportional to the change in distance between source and receiver. Since the radar moves during the transmission of the subsequent pulses, the distance $R(t_i)$ changes in t_i and thus the phase of the echo signal changes in t_i. This induces a Doppler shift in the

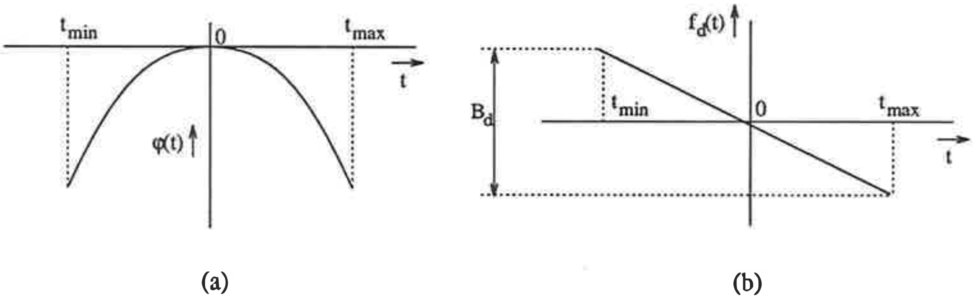


Figure 2.4: The phase history $\varphi(t)$, (a), and the Doppler frequency $f_d(t)$, (b), of a point target.

azimuth direction of the echo signal. Thus, since the echo signal is discrete in the azimuth direction, the sampling frequency f_{prf} must satisfy the Nyquist criterion to prevent aliasing. At this point, though, we are only interested in the Doppler shift in azimuth. We simply assume that f_{prf} satisfies the Nyquist criterion. To simplify the Doppler analysis we consider the echo signal time-continuous in azimuth, thereby omitting the index i, i.e., $t \equiv t_i$ and $x \equiv x_i$.

Since the Doppler shift is completely determined by the phase history of the point target $\varphi(t) = -\omega_c 2R(t)/c$, it is sufficient to use the simplified echo signal

$$e(t) = \exp(-j\omega_c 2R(t)/c) \tag{2.6}$$

Then the Doppler frequency can be derived as the derivative of the phase history

$$f_d(t) = \frac{1}{2\pi}\frac{\mathrm{d}}{\mathrm{d}t}\varphi(t) = -\frac{2}{\lambda}\frac{\mathrm{d}}{\mathrm{d}t}R(t) \tag{2.7}$$

In general $R_0 \gg |x|$, for all $x_{min} \leq x \leq x_{max}$, then, from figure 2.3, we have

$$R(t) = \sqrt{R_0^2 + x^2} = R_0 + \frac{v^2t^2}{2R_0} - \frac{v^4t^4}{8R_0^3} + \cdots \tag{2.8}$$

Thus we can approximate equation (2.8) as

$$R(t) = R_0 + \frac{v^2t^2}{2R_0} \tag{2.9}$$

up to an error term $\mathcal{O}(v^4t^4/8R_0)$. This means that the phase $\varphi(t)$ is approximately quadratic in t, and the Doppler frequency approximates to

$$f_d(t) = -\frac{2v^2t}{\lambda R_0} \tag{2.10}$$

The phase and the Doppler frequency of the echo signal of the point target are shown in figure 2.4.a and 2.4.b, respectively. The corresponding *Doppler bandwidth* B_d is the difference between the minimum and maximum of the Doppler frequency

$$B_d = |f_d(t_{max}) - f_d(t_{min})| = \frac{2v^2(t_{max} - t_{min})}{\lambda R_0} = \frac{2vL_{illum}}{\lambda R_0} = \frac{2v}{D} \qquad (2.11)$$

2.3 Pulse Compression

In the previous section we neglected the pulse width τ_p. This is allowed if we can accomplish a sufficiently small pulse width τ_p, so that $p(t)$ approaches a dirac signal. In practice, however, the minimum feasible pulse width is relatively large, due to physical limitations in antenna and transmitter components. Still, there are a number of design considerations which must be taken into account [Sko85]. The two most important considerations in SAR are:

- The signal energy must be high enough to detect targets (resolution cells) at the specified range and in the presence of noise.
- The range resolution must meet its specification. Often it is specified that the range resolution must be equivalent to the azimuth resolution.

The combination of these two items implies a very short pulse with high peak power. This is not feasible in practical radar systems.

The key solution to this problem is to transmit a pulse with a specific waveform, such that a long pulse can be transmitted with high energy but low peak power. After receiving the echo the resolution and signal-to-noise ratio can be enhanced by means of signal processing: the so-called pulse compression. In radar, pulse compression is generally performed by means of a so-called *matched filter*, i.e., the received echo signal is correlated by a replica of the transmitted pulse. The principle of matched filtering has its origin in information theory [Nor63], and is used to detect a known signal in the presence of noise.

We briefly summarize the principle of matched filtering for our case. For a concise derivation of the matched filter principle, we refer to [Pap84]. Let $p(t)$ be the transmitted pulse and let $n(t)$ be additive white noise with zero-mean. Let $e(t) = p(t - t_p) + n(t)$ be the received echo signal of the shifted pulse, reflected by some point target in the antenna beam. Note that the distance between the antenna and the point target is proportional to the shift t_p. The objective is now to detect the pulse echo within the signal $e(t)$ and to localize it. This can be accomplished by filtering the received signal with the matched filter having the impulse response $h(t) \triangleq p^*(-t)$. Let $y(t) = h(t) * p(t - t_p) + h(t) * n(t) \triangleq y_p(t) + y_n(t)$ be the output of the matched filter. Then the matched filter maximizes the ratio between $|y_p(t_p)|^2$ and the expected noise power $\mathsf{E}|y_n(t)|^2$ at the output of the matched filter. The peak is located at t_p.

However, the matched filter does not enhance resolution per se. Resolution enhancement by matched filtering can be accomplished by transmitting a waveform with an autocorrelation function that approximates a dirac function. Many waveforms are known [Del70], but commonly used in SAR is the linear frequency modulated (FM) pulse, also known as the *chirp* signal. Let $p(t)$ be a linear frequency modulated pulse with frequency varying linear in time, $\alpha t/\pi$, and pulse width τ_p. Then we define

$$p(t) = \exp(j\alpha t^2), \qquad \text{for } -\tfrac{1}{2}\tau_p \leq t \leq \tfrac{1}{2}\tau_p \tag{2.12}$$

and zero otherwise. α is called the chirp-rate. The bandwidth of a linear FM pulse is $B_p = \alpha\tau_p/\pi$. Matched filtering of the chirp results in a sinc-function

$$\int p^*(-\tau)p(t-\tau)\mathrm{d}\tau \simeq \operatorname{sinc}(2\pi B_p t) \tag{2.13}$$

Two examples are given in figure 2.5. Observe that the waveform is the same as in figure 2.7, which is the reason why the matched filter is also commonly used for azimuth compression. Furthermore we can see that for larger time-bandwidth product $\tau_p B_p$ the spectrum of the linear FM pulse becomes more and more rectangular.

It remains to derive the range resolution after the received pulse echo has been processed by the matched filter. Here we use the fact that for large time-bandwidths $\tau_p B_p$, the power spectrum is approximately rectangular with bandwidth B_p [Del70]. The resolution is then defined as the half-power width of the sinc, which approximates $1/B_p$. The range resolution, say ρ_r, is then the *two-way* spatial half-power width of the compressed pulse

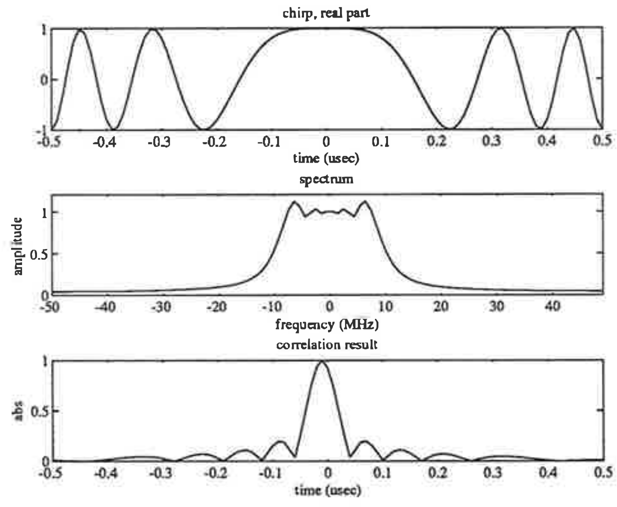
$$\rho_r = \tfrac{1}{2}c/B_p = \frac{\pi c}{2\alpha\tau_p} \tag{2.14}$$

Observe that the ratio between the width of the pulse, τ_p, and the width of the compressed pulse, $1/B_p$, equals the time-bandwidth product $\tau_p B_p$. Therefore the time-bandwidth product of the pulse is also called the *pulse compression ratio.*

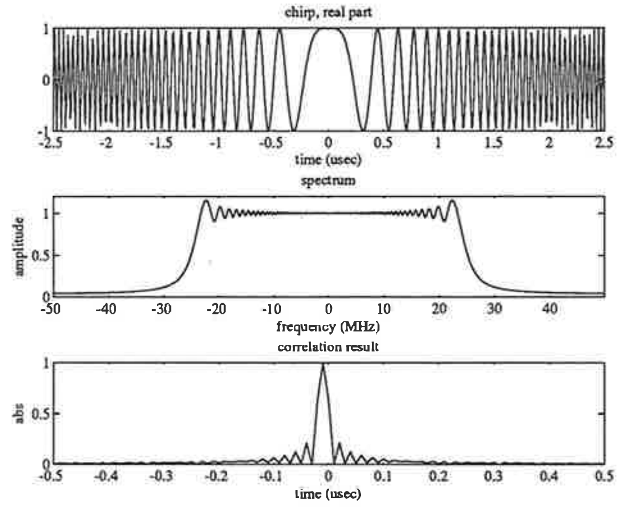
2.4 Azimuth Compression

Before we proceed with the exploitation of the Doppler bandwidth in SAR imaging we introduce the *optimum azimuth resolution.* Consider the situation as shown in figure 2.6 where there are two adjacent point targets with equal nominal range R_0. Let their x-coordinates be 0 and $\delta x \equiv v\delta t$, respectively. We assume that throughout the time that the point targets are within the antenna beam, the azimuth echo signal is acquired. The optimum azimuth resolution[1] of a side-looking imaging radar is the minimum spatial distance δx between two adjacent point targets, for which they still can be discriminated within the echo signal in azimuth direction. We

[1]We assume that resolution of a radar image is user specified within certain boundaries, one of them being the optimum resolution, that is, it is the best one can get given the system parameters.



(a)



(b)

Figure 2.5: Two examples of linear FM pulse: (a) $\tau_p = 1$ μsec, $\alpha = 6.28 \times 10^{13}$ and $\tau_p B_p = 20$ and (b) $\tau_p = 5$ μsec, $\alpha = 3.14 \times 10^{13}$ and $\tau_p B_p = 250$.

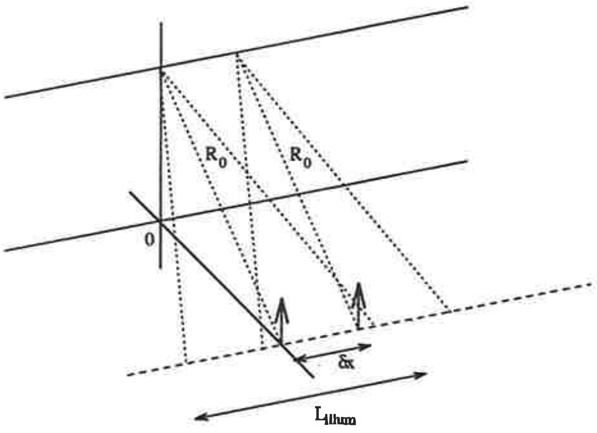


Figure 2.6: Two adjacent point targets with equal nominal range R_0 and at relative distance δx in azimuth direction. L_{illum} is the length of the antenna beam in azimuth at nominal range R_0.

assume that this is the spatial half-power beamwidth of the antenna (either a real aperture or a synthetic aperture antenna) on the ground. We usually denote the optimum azimuth resolution by ρ, whereas the azimuth resolution of the SAR image is denoted by δx. The difference between ρ and δx is that ρ is a SAR acquisition system parameter and δx is a SAR reconstruction system parameter. I.e., in our case ρ is given, and δx is specified (either by the designer of the reconstruction system or by the end-user), with the condition that $\delta x \geq \rho$.

A radar system that images the 2D radar echo signal without resolution enhancement in azimuth is called a *real aperture radar* or a *side-looking airborne radar (SLAR)* system. Let ρ be the optimum azimuth resolution of this system at nominal range R_0, then, according to the definition of the optimum azimuth resolution,

$$\rho = L_{illum} = R_0\lambda/D \tag{2.15}$$

Despite the relatively narrow azimuth beam $\beta = \lambda/D$, in SLAR systems ρ can be quite bad, typically $\mathcal{O}(100)$ m for airborne systems. This is caused by the relatively large nominal range R_0. The resolution in range can be much better, typically $\mathcal{O}(1)$ m. Moreover, ρ depends on system parameters as wavelength and range, which is not desirable.

Optimum azimuth resolution of SAR

Using SAR we can achieve a much higher azimuth resolution, typically $\mathcal{O}(1)$ m for airborne systems, which is also independent of the wavelength and the nominal range. There are many viewpoints from which the SAR principle can be derived, all yielding "more or less" the same expressions. We do not wish to be complete here, in that we simply adopt one viewpoint which is illustrative in our opinion. For other viewpoints we refer to, e.g., [Cut70, Har70, Hov80, CM91].

Substitute equation (2.9) into equation (2.6), then the echo signal of a single point target is

$$e(t) = \exp\left(-j\frac{4\pi}{\lambda}\frac{v^2t^2}{2R_0}\right) \tag{2.16}$$

We assume that $e(t)$ is defined in the interval (t_{min}, t_{max}) and is zero otherwise. In [Del70], it was shown that if the time-bandwidth T_aB_d of $e(t)$ is sufficiently large (say > 10, see [CM91]), then the power spectrum of $e(t)$ approximates a rectangular with bandwidth B_d. Note that the power spectrum of $e(t)$ is the Fourier transform of $r(t) = e^*(-t) * e(t)$. Thus, conversely, if $r(t)$ has a rectangular real Fourier transform with bandwidth B_d then $r(t) \simeq \text{sinc}(2\pi B_d t)$. We consider the width of the main lobe of $|r(t)|^2$, which is approximately $1/B_d$, as the optimum temporal azimuth resolution. Then the optimum azimuth resolution of the SAR, say ρ_a, approximates to $\rho_a = v/B_d$. Using equation (2.11) for the Doppler bandwidth, we obtain

$$\rho_a = v/B_d = D/2 \tag{2.17}$$

which is the usual expression for the optimum azimuth resolution of SAR. Better resolution in azimuth can thus be accomplished by filtering the echo signal with

$$a(t) \equiv e^*(-t) = \exp\left(j\frac{4\pi}{\lambda}\frac{v^2t^2}{2R_0}\right) \tag{2.18}$$

We refer to $a(t)$ as the impulse response of the *1D azimuth filter*.

An advantage of SAR over SLAR is that ρ_a is independent of the nominal range and wavelength. It depends only of the length of the physical antenna. The latter should be no surprise: the smaller the antenna, the larger the Doppler bandwidth, see equation (2.11), and thus the smaller ρ_a. The duration of the time interval needed to acquire echo signal, $T_a = t_{max} - t_{min}$, is called the *synthetic aperture time* or simply *aperture time*. For convenience, we assume that $t_{min} = -\frac{1}{2}T_a$ and $t_{max} = \frac{1}{2}T_a$. Corresponding to this synthetic aperture time the *synthetic aperture length* is defined as

$$L_a = x_{max} - x_{min} = vT_a = R_0\lambda/D \equiv L_{illum} \tag{2.19}$$

We can also relate the synthetic aperture length with the optimum azimuth resolution ρ_a. From equation (2.11) follows that $T_a = \frac{1}{2}B_d\lambda R_0/v^2$. Thus, with $\rho_a = v/B_d$, we have

$$L_a = vT_a = \tfrac{1}{2}B_d\lambda R_0/v = \tfrac{1}{2}\lambda R_0/\rho_a \tag{2.20}$$

If we compare the optimum azimuth resolution ρ of SLAR with the optimum azimuth resolution ρ_a of SAR, we have achieved a resolution improvement factor ρ/ρ_a. Using $\rho =$

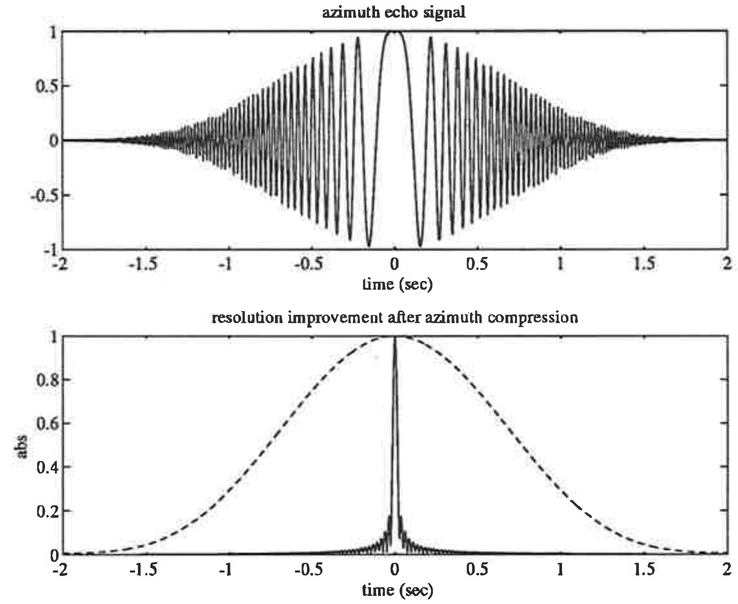


Figure 2.7: Simulated azimuth echo signal (real part) with antenna pattern and its real aperture resolution (dashed line) and the synthetic aperture resolution. The used radar parameters are: $\lambda = 0.057$ m, $v = 100$ m/sec, $R_0 = 10$ km, and $T_a = 1$ sec.

$L_{illum} = vT_a$ and $\rho_a = v/B_d$, it follows that this improvement is equal to the time-bandwidth product T_aB_d. This resolution improvement factor can be up to $\mathcal{O}(10^3)$ in practical SAR systems. It is illustrated in figure 2.7, where we have used simulated azimuth echo data.

As we have mentioned implicitly, obtaining the autocorrelation signal from the echo signal yields the optimum azimuth resolution ρ_a. Observe that, if we model the point target by a dirac function, then the echo signal $e(t)$ is, in fact, the impulse response of the simplified SAR acquisition system. To accomplish a high resolution SAR image, the received echo signal should thus be correlated by the known impulse response of the SAR acquisition system. The correlation of the received echo signal with the known impulse response $e(t)$ is called *azimuth compression*. A useful side-effect of the azimuth compression is that it is equivalent to *matched filtering*, which is an optimum detector of a known waveform (such as the impulse response) in the presence of noise, see section 2.3.

Depth of focus

The *depth of focus* is a phenomenon that results from the range dependence.As we have shown above, for each nominal range R_0 we should use a different azimuth filter. Consider the azimuth echo signal corresponding to nominal range R_0. Suppose that the azimuth filter is optimal for nominal range $R_0 + \delta R$, i.e., we have a mismatch δR in nominal range. It is useful to know the effect of the mismatch in nominal range on the reconstructed signal. The depth of focus is the relation between the mismatch in nominal range and the azimuth resolution.

From equation (2.16) we see, that the mismatch δR in nominal range corresponds to a mismatch $\delta\varphi$ in the phase of the impulse response of a point target at nominal range R_0 and the azimuth filter. Let

$$\varphi(t) = \frac{4\pi}{\lambda}\frac{v^2t^2}{2R_0} \tag{2.21}$$

be the phase history of a point target at nominal range R_0. Then the phase mismatch is

$$\delta\varphi = -\frac{4\pi}{\lambda}\frac{v^2t^2}{2(R_0+\delta R)} + \frac{4\pi}{\lambda}\frac{v^2t^2}{2R_0} \tag{2.22}$$

If $|\delta R| \ll R_0$, then the phase mismatch approaches $\delta\varphi = -\varphi(t)\delta R/R_0$. Observe that $|\varphi(t)|$ is maximum for $t = t_{min/max} = \pm\,\frac{1}{2}T_a$. Thus

$$|\delta\varphi| = \varphi(\pm\,\tfrac{1}{2}T_a)\frac{|\delta R_0|}{R_0} = \frac{\pi v^2 T_a^2}{2\lambda R_0^2}|\delta R_0| \tag{2.23}$$

From equation (2.20) we have $T_a = \frac{1}{2}\lambda R_0/v\rho_a$. Substitute T_a in equation (2.23), then we obtain

$$|\delta\varphi| = \frac{\pi\lambda}{8\rho_a^2}|\delta R_0| \tag{2.24}$$

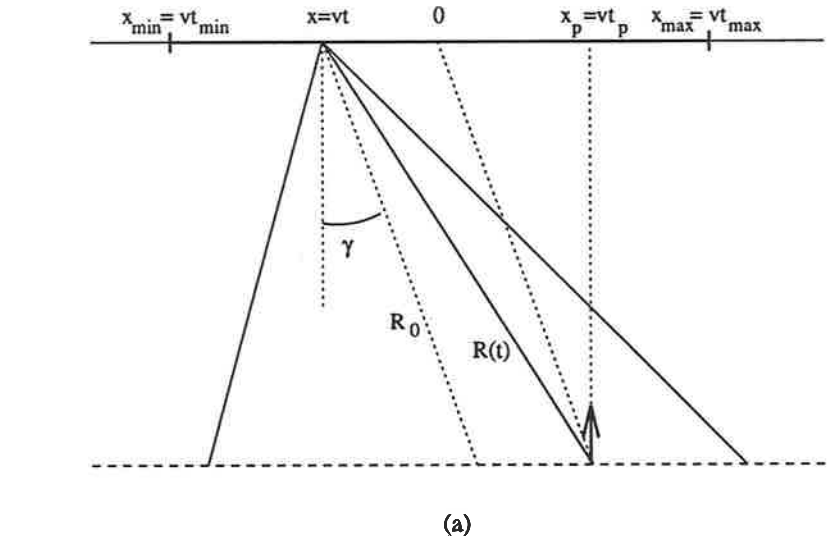
If we allow a maximum phase mismatch of $\pm\,\pi/4$ [CM91], then we have

$$|\delta R| < \frac{2\rho_a^2}{\lambda} \tag{2.25}$$

From the latter term we derive what is called in the literature the depth of focus

$$\Delta R_{dof} \triangleq 4\rho_a^2/\lambda \tag{2.26}$$

The depth of focus is a criterion for the maximum mismatch in nominal range of the azimuth filter and the azimuth echo signal that is allowed, given the wavelength and optimum azimuth resolution. In practice, this means that we can use one azimuth filter assigned to nominal range R_0 to process the azimuth echo signals assigned to an interval of the nominal range $(R_0 - \frac{1}{2}\Delta R_{dof}, R_0 + \frac{1}{2}\Delta R_{dof})$. For example, if $\lambda = 0.057$ m (the wavelength corresponding to C-band) and $\rho_a = 2$ m, then the depth of focus is $\Delta R_{dof} = 140$ m.



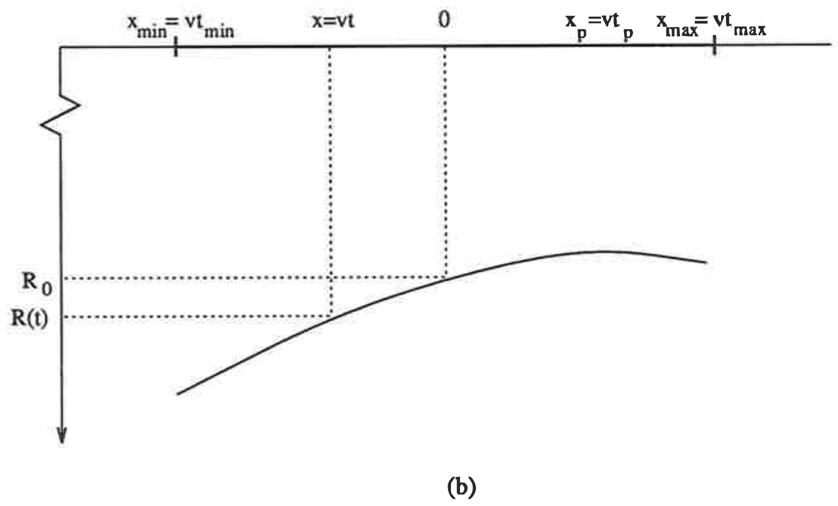


Figure 2.8: The geometry in azimuth of the SAR system with non-zero squint angle (a) and the non-constant range $R(t)$ of a single point target as function of the time (b).

Range migration

Until here we have used the simplified echo signal $e(t)$, thereby neglecting the shift in $p(t' - 2R(t)/c)$. It allowed us to analyze the Doppler frequency of an echo signal in azimuth of a single point target along a straight line parallel to the x-axis. However, in general this is not the case, especially if the squint angle is non-zero. In this section we derive the general descriptions of the SAR acquisition system A and SAR reconstruction system R for non-zero squint angle γ.

Consider the geometry in azimuth of the SAR system in figure 2.8.a. Assume between $t_{min} = -\frac{1}{2}T_a$ and $t_{max} = \frac{1}{2}T_a$ a point target with x-coordinate x_p is illuminated by the antenna beam. We assume that at $t = 0$ the distance between the antenna and point target is the nominal range R_0, thus $x_p = R_0 \sin\gamma$. Let $x = vt$ and define $t_p = x_p/v$. From figure 2.8.a we can derive the extended version of equation (2.8)

$$R(t) \; = \; \sqrt{R_0^2 \cos^2\gamma + (x_p - x)^2} \; = \; \sqrt{R_0^2 - 2xR_0 \sin\gamma + x^2} \tag{2.27}$$

Let $|\sin\gamma| \ll 1$, then we have $R_0 \ll \sqrt{|2xR_0 \sin\gamma - x^2|}$. Thus $R(t)$ approximates to

$$R(t) \; = \; R_0 - vt\sin\gamma + \frac{v^2t^2}{2R_0} \; \triangleq \; R_0 + \Delta R(t) \tag{2.28}$$

In figure 2.8.b $R(t)$ is shown as function of t. The linear decline of $R(t)$, called the *range walk*, depends on the squint angle γ, and the non-linear decline of $R(t)$, called the *range curvature*, depends on the aperture time T_a. The range migration is illustrated in figures 2.8.a and 2.8.b.

Substitution of $R(t)$ into equation (2.5) gives

$$e(t', t) \; = \; p\,(t' - 2\Delta R(t)/c) \exp\left(-j4\pi \Delta R(t)/\lambda\right) \tag{2.29}$$

To simplify the notation, we have applied a shift $t' - 2R_0/c \leftarrow t'$ in equation (2.29). This does not affect the analysis, as the location of the origin of t' is only a matter of reference. Equation (2.29) is the 2D echo signal of a single point target. If we model the point target by a dirac function and if we neglect the atmospheric attenuation, system losses, etc., then $e(t', t)$ is, in fact, the 2D impulse response of the SAR acquisition system A.

At this point we obtain the 2D impulse response of the SAR reconstruction system R. The first step of the reconstruction is the *range compression*.[2] Let $g(t', t)$ be the 2D echo signal, correlated with a replica of the transmitted pulse $p(t')$. Assume that $p(t')$ is the chirp signal with bandwidth B_p, as described in section 2.3.

$$\begin{aligned} g(t', t) \; &= \; \int p^*(-\tau) e(t' - \tau, t) \mathrm{d}\tau \\ &\simeq \; \mathrm{sinc}\,(2\pi B_p(t' - 2\Delta R(t)/c)) \exp\left(-j4\pi \Delta R(t)/\lambda\right) \end{aligned} \tag{2.30}$$

[2]In SAR it is customary to use the term range compression for pulse compression.

Observe that the peak of the position of the sinc-peak varies in t, due to the range migration. Obviously, this shift can be eliminated by convolving $g(t', t)$ over t' with a dirac signal $\delta(t' + 2\Delta R(t)/c)$. Let $g'(t', t)$ be the echo signal after the range migration compensation, then

$$\begin{aligned} g'(t', t) &= \int \delta(\tau + 2\Delta R(t)/c) g(t' - \tau, t) \mathrm{d}\tau \\ &= g(t' + 2\Delta(t)/, t) \simeq \operatorname{sinc}(2\pi B_p t') \exp(-j4\pi \Delta R(t)/\lambda) \end{aligned} \tag{2.31}$$

Another side-effect of the range migration we must take into account is that the Doppler spectrum (i.e., the spectrum in azimuth direction) is shifted. This frequency offset, say f_{dc}, is called the *Doppler centroid*, as it is the center of the Doppler spectrum. The Doppler centroid is the constant term in the derivative of the phase history of the point target. Let $\varphi(t) = -4\pi\Delta R(t)/\lambda$ be the phase history, see equation (2.30), then

$$\frac{1}{2\pi}\frac{\mathrm{d}}{\mathrm{d}t}\varphi(t) = -\frac{2}{\lambda}\frac{\mathrm{d}}{\mathrm{d}t}\Delta R(t) = \frac{2v\sin\gamma}{\lambda} - \frac{2v^2 t}{\lambda R_0} \tag{2.32}$$

Thus $f_{dc} = 2v\sin\gamma/\lambda$. The second term is recognized as the Doppler frequency shift $f_d(t)$, see section 2.2. The derivate of the Doppler frequency shift $f_d(t)$ is also called the *Doppler rate* $f_R = -2v^2/\lambda R_0$. Note that the Doppler bandwidth is still $f_d(t_{max}) - f_d(t_{min}) = B_d$.

A consequence is that the azimuth filter should also contain this frequency offset. Then, using the azimuth filter $a(t)$ derived in section 2.4, the 1D azimuth filter in this case is $a(t)\exp(2\pi f_{dc} t)$. Thus, convolving $g'(t', t)$ with this azimuth filter over t, gives the reconstructed signal $r(t', t)$

$$r(t', t) = \int a(\tau)\exp(2\pi f_{dc}\tau) g'(t', t - \tau)\mathrm{d}\tau \simeq \operatorname{sinc}(2\pi B_p t')\operatorname{sinc}(2\pi B_d t) \tag{2.33}$$

Observe that we have derived the sinc functions $\operatorname{sinc}(2\pi B_p t')$ and $\operatorname{sinc}(2\pi B_d t)$ earlier in sections 2.2 and 2.4, respectively.

Of course, we can also combine the compensation for the range migration and the azimuth compression. Then we should use the 2D azimuth compression filter

$$\underline{a}(t', t) = a(t)\exp(2\pi f_{dc} t)\delta(t' + 2\Delta R(t)/c) \tag{2.34}$$

Obviously, the 2D impulse response of the SAR reconstruction system is thus $p^*(-t)\underline{a}(t', t)$. Notice that this impulse response represents, in fact, a 2D matched filter.

We have stated in section 2.1, that in the ideal world R would be the inverse of A. The reconstruction system R as we have determined, approaches the inverse A^{-1}, if B_p and B_d both approach infinity. Indeed, in designing high resolution SAR systems, bandwidth increment has in general a high priority. The impulse response of the reconstruction system, the 2D matched filter, is optimum in terms of signal-to-noise ratio, however, it does not minimize the square error of the desired signal (the reflectivity map of the imaged surface) and the reconstructed SAR image. In appendix B we propose an alternative filter, that performs suboptimum in terms of signal-to-noise, but yields the minimum mean square error.

CHAPTER 3

SAR PROCESSOR SPECIFICATIONS

3.1 Introduction

Typical developments of the last decade in the field of SAR processing can be divided into two areas: two-dimensional high-resolution SAR processing algorithms, e.g., [Cen88, FS90, CPR91, RRB$^+$94], and dedicated real-time SAR processing hardware, e.g., [Mor92, DJCM92, Bie93]. In the first area the ongoing progress in the general purpose computer hardware is exploited by development of complex two-dimensional SAR processing algorithms. The objective is to achieve high-resolution SAR images, whereas processing speed is of minor importance. In the second area dedicated hardware is developed for real-time on-board airborne SAR processing. Although it has lower priority in real-time on-board SAR processing, image quality becomes more and more important. For example, in [Mor92, Bie93] an attempt is made to balance between algorithm and architecture design, which should lead to an optimal SAR processor.

The objective of this chapter is to give a brief introduction in the design aspects of a real-time SAR processor. We translate the SAR acquisition system parameters determined in the previous chapter, to SAR processing parameters. We do not intend to design a generic real-time SAR processor, in that, we restrict ourselves to antenna systems having small squint angles and narrow azimuth antenna beamwidth (both in the order of a few degrees). The justification of this restrictions is in the fact that the work presented in this thesis is strongly related to the PHARUS system. PHARUS stands for Phased Array Universal SAR and is a fully polarimetric C-band airborne SAR with an active antenna array. The PHARUS system employs yaw steering to minimize the squint angle, and has a small azimuth antenna beamwidth in the order of 2°. More details about the PHARUS system are given in section 3.3.

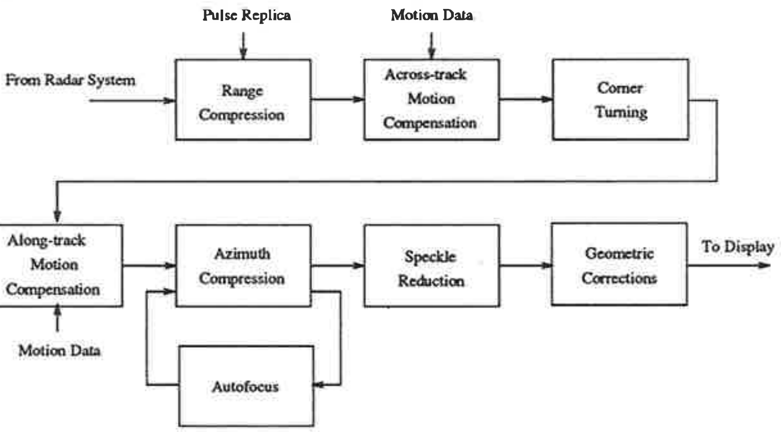


Figure 3.1: Typical airborne SAR processing chain.

Due to the small squint angle and the narrow beam, range migration effects are limited. As a consequence, we can perform the basic SAR processing algorithms (range and azimuth compression) using 1D fast convolution techniques. The azimuth compression approach is related to the so-called Hybrid Correlation approach [WLJ82]. It performs a 2D convolution, hereby utilizing fast convolution in azimuth direction and time domain convolution in range direction. This approach has the advantage that data flows and memory management remain relatively simple, so that we can focus on the design of *small* and *high speed* SAR processing hardware. Since the approach is straight-forward, the processing is robust and the effects on the SAR image quality are well-understood.

Our lead is a typical airborne SAR processing chain, shown in figure 3.1. In section 3.2 we translate the SAR parameters to SAR processing parameters. We describe them per functional block, except for the motion compensation and the autofocusing, which are beyond the scope if this thesis. We simply assume that motion compensation and/or autofocusing is performed and that it satisfies our needs. Nevertheless, in appendix C a brief description of motion compensation and autofocusing aspects are given. In section 3.3 we show the usefulness of the processing parameters by a design example: a real-time SAR processor for PHARUS. The result of this section is a set of specifications for real-time SAR processing hardware, such as range and azimuth compression data rates, number of looks, filter sizes and data reduction factors. In section 3.4 we show that our approach does not only apply to the PHARUS system. We specify a real-time SAR processor for the ERS-1 satellite SAR system.

3.2 SAR Processing System Parameters

In this section we assume that the raw SAR data is sampled in range and azimuth. we assume that sampling frequencies in range and azimuth, say f_r and f_a, respectively, satisfy the Nyquist criterion. Thus, let B_p be the pulse bandwidth and let B_d be Doppler bandwidth, then $f_r > B_p$ and $f_a > B_d$.[1] Note that in section 2.2 we have interpreted the pulse repetition frequency f_{prf} as the azimuth sample frequency. In many airborne SAR systems, however, f_{prf} is substantially larger than the Doppler bandwidth. The azimuth sample frequency f_a is then obtained by bandpass filtering and decimation of the azimuth echo signals, i.e., f_{prf} is then a multiple of f_a. It is assumed that the reception time per pulse echo is limited and thus the length of the pulse echo is limited. We denote the length of the discrete pulse echo by N_r. In practice, N_r is $\mathcal{O}(10^3)$ up to $\mathcal{O}(10^4)$.

Range compression

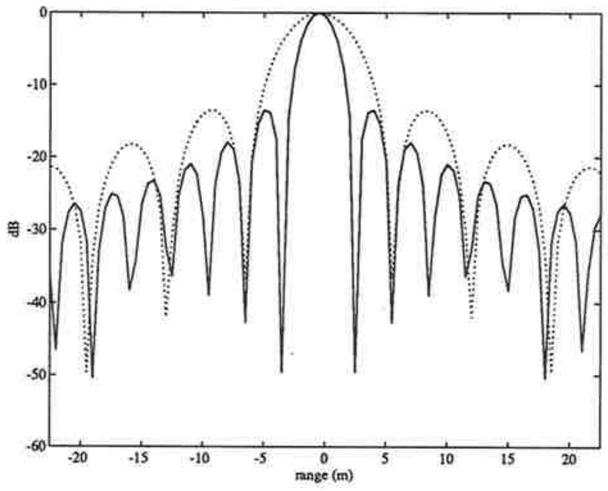
Let the transmitted pulse be the chirp signal as defined in equation (2.12), with pulse width τ_p and bandwidth B_p. Let t_r be the sampling period in range, then the length of the discrete pulse replica is $N_p = \tau_p/t_r$. The advantage of discrete range compression is that the pulse shape can easily be modified. For example, we can use a shorter pulse replica. A property of the chirp signal is that its bandwidth is proportional to its length. Recall that the optimum range resolution ρ_r is inverse proportional to the pulse bandwidth, see equation (2.14). Thus, a shorter pulse replica yields a degraded resolution. In this way we can specify a suboptimum range resolution, say $\delta r \geq \rho_r$, which is accomplished by correlating the received pulse echoes with a pulse replica with length reduced with a factor $\delta r/\rho_r$. This is illustrated in figure 3.2.a. Moreover, since the bandwidth of the pulse replica is also reduced with a factor $\delta r/\rho_r$, the range sample frequency might also be reduced after the range compression. Thus, if we require a suboptimum range resolution, data reduction (and thus data rate reduction) can be achieved.

Another important advantage of discrete range compression is that it allows the application of weighting. By windowing the pulse replica, the peak side-lobe ratio (PSLR) can be improved. The PSLR is defined as the ratio of the largest side-lobe peak to the main-lobe peak [CM91] and is usually expressed in dB. Some examples of pulse compression and weighting are shown in figure 3.2.b. Clearly, reduction of side-lobes results in an increase of the resolution. The increment factor of the resolution for different types of windows can be found in [Sko85].

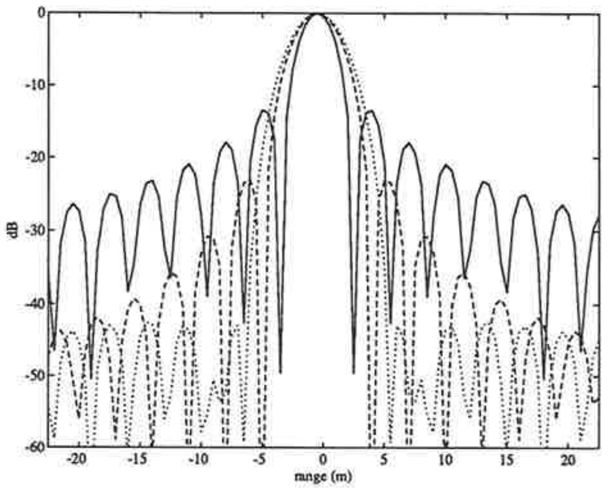
Azimuth compression

Given the 1D azimuth filter as defined in equation (2.18). Let t_a be the sampling period in azimuth, and observe that the filter length is the aperture time T_a, then the length of the discrete 1D azimuth filter is $N_a = T_a/t_a$. Equation (2.20) gave the relationship between the aperture

[1]Note that the sample frequencies are defined for complex signals.



(a)



(b)

Figure 3.2: Pulse compression with a pulse replica length $\tau_p = 5$ μsec (straight line) and $\tau_p = 2.5$ μsec (dotted line) (a), and the effect of weighting: uniform weighting, PSLR=13 dB (straight line), cosine weighting, PSLR=23 dB (dashed line) and hamming weighting, PSLR = 43 dB (dotted line) (b).

time T_a and the optimum azimuth resolution ρ_a. If we specify a suboptimum azimuth resolution $\delta x \geq \rho_a$, then the aperture time is reduced to $T_a = \frac{1}{2}\lambda R_0/\delta x v$. Thus a factor $\delta x/\rho_a$ reduction of the discrete azimuth filter length is obtained. Similar to the range compression, this also yields a factor $\delta x/\rho_a$ bandwidth reduction after azimuth compression, and thus data reduction can be achieved.

The update rate of the azimuth filter depends on the depth of focus. Given the suboptimum azimuth resolution δx, then the depth of focus is $\Delta R_{dof} = 4\delta x^2/\lambda$. Let t_r be the sampling period in range, then the sampling distance in range is $dr = \frac{1}{2}ct_r$. Then updating of the azimuth filter is only required for every $N_{dof} \triangleq \Delta R_{dof}/dr$ subsequent range bins.

As we have mentioned in section 2.4, range migration compensation might be required. It depends on the maximum decline of $R(t) = R_0 + \Delta R(t)$. Let δr be the suboptimum range resolution. Then a sufficient criterion is that compensation is necessary if the maximum value of $\Delta R(t)$ over the interval $(-\frac{1}{2}T_a, \frac{1}{2}T_a)$ exceeds $\delta r/4$ [CM91]. Thus

$$\tfrac{1}{2}vT_a|\sin\gamma| + \frac{v^2T_a^2}{8R_0} \geq \frac{\delta r}{4} \tag{3.1}$$

Substitute $T_a = \frac{1}{2}\lambda R_0/v\delta x$ into equation (3.1), then we obtain that range migration compensation is necessary if

$$\frac{\lambda R_0}{\delta x}\left(|\sin\gamma| + \frac{\lambda}{8\delta x}\right) \geq \delta r \tag{3.2}$$

This expression is called the *range migration criterion*. Given that λ and γ are system parameters and that R_0 is bounded by some maximum R_{max}, we conclude that the range migration criterion depends on δr and δx.

The range migration compensation can be performed in various ways. We only consider the straight forward method, that is, we perform the azimuth compression with the discrete version of the 2D azimuth filter, see equation (2.34), which performs range migrations correction and azimuth compression all at once. Let ΔR_{mig} be the maximum decline of $R(t)$ over the interval $(-\frac{1}{2}T_a, \frac{1}{2}T_a)$ and let dr be the sample distance in range, then the size of the discrete 2D azimuth filter in range is $N_{mig} = \Delta R_{mig}/dr$.

Corner turning memory

The corner turning memory is a device to store the range compressed data set. In general, azimuth compression is performed on batches of azimuth echo samples, whereas the acquisition of the azimuth echo samples is a continuous process at the rate of the azimuth sample frequency. The function of the corner turning memory is briefly addressed here, where we consider the following preliminaries. A *range line* refers to a discrete range compressed pulse echo. An *azimuth line* refers to a the set of azimuth echo samples that corresponds to a specific range.

Thus, the i^{th} components of subsequent range lines forms the i^{th} azimuth line. We have $N_r - N_p$ azimuth lines, which is the number of effective samples that results from the range compression operation. In general we assume that an azimuth line is finite, which is then referred to as an *azimuth batch*. The azimuth compression operates on azimuth batches, thereby using an overlap-discard approach. The overlap is determined by the azimuth filter length N_a. The function of the corner turning memory is then: collect N_b range lines and write all $N_r - N_p$ azimuth batches sequentially to the azimuth compression.

It is now obvious, that during the writing of the azimuth batches to the azimuth compression, new range lines keep coming. Thus the corner turning memory should be dual-ported, and have sufficient storage capacity to store the new range lines while writing the batches. This implies a "ping-pong" memory structure: range lines are written in the "ping"-part of the memory and azimuth batches are read from the "pong"-part of the memory. When the "ping"-part is full and the "pong"-part is empty, they interchange the "ping" and "pong" functionality.

However, we must take into account the overlap N_a. Therefore, from hardware point of view it is useful to use the concept of *banking*. This concept is explained by the following example. Assume that the overlap is 50 %, and thus $N_b = 2N_a$, and consider the corner turning memory with three banks. Each bank has size $N_r - N_p \times N_a$. Assume that at a certain moment banks 1 and 2 have been filled, and bank 3 is still empty. At this point we start writing range lines into bank 3, and in the mean we read the azimuth batches from banks 1 and 2. Obviously we have a limited time to read the azimuth batches, namely N_a/f_a, which is the acquisition time of N_a range lines. When the banks 1 and 2 have been read and bank 3 has been filled, the range lines are written into bank 1 and the azimuth batches are read from bank 2 and 3. This concept can readily be extended to more banks.

Obviously the output data rate is higher than the input data rate, thus we have a mismatch between input and output data rate induced by the overlap. This effect is best described by the *utilization* of the corner turning memory U, which is the ratio of the required input data rate and the required output data rate. Observe that the utilization is completely determined by the azimuth batch length N_b and the length of the overlap N_a

$$U = \frac{N_b - N_a}{N_b} \tag{3.3}$$

The measure of the utilization is in percents %.

The utilization is an important design parameter. For example, if the utilization approach 100 % we have large N_b relative to N_a. In general this means that the acquisition time N_b/f_a (which is the time required to acquire N_b pulse echoes) can be large. If we use the rough estimation that azimuth compression of the azimuth batch requires N_bU/f_a seconds, then we have a *latency*

$$\mathcal{L} = N_b(1+U)/f_a \tag{3.4}$$

The latency $\mathcal{L}$ is a measure for the time between the first appearance of a point on the ground within the antenna beam and the imaging of that point on the display. Conversely, if we require a small latency, the utilization will be small. In that case N_b is small relative to N_a, but then the efficiency of the processing drops dramatically.

Speckle reduction

A surface area within a SAR image with a constant backscatter coefficient σ_0 appears as a noisy signal with mean $\overline{\sigma}_0$ [UMF82]. This noise is referred to as *speckle noise*. Let z be the intensity of a SAR image pixel with mean $\overline{z}$ and variance ν^2. To improve the visual appearance of an image, it is desirable to cluster the observed intensities z closer around $\overline{z}$, which is equivalent to reducing the standard deviation. If we assume that we have N_l independent realizations z_n, $n = 0, \cdots, N_l - 1$ of an image pixel then the mean of the average $z_{N_l} = (1/N_l)\sum_{n=0}^{N_l-1} z_n$ is unchanged

$$\overline{z}_{N_l} = \mathsf{E}\frac{1}{N_l}\sum_{n=0}^{N_l-1} z_n = \frac{1}{N_l}\sum_{n=0}^{N_l-1} \mathsf{E}z_n = \overline{z} \tag{3.5}$$

whereas the standard deviation ν_{N_l} is reduced by a factor $\sqrt{N_l}$. This can be derived as follows

$$\nu_{N_l}^2 = \mathsf{E}(z_{N_l} - \overline{z}_{N_l})^2 = \mathsf{E}(z_{N_l} - \overline{z})^2 = \frac{1}{N_l^2}\sum_{n=0}^{N_l-1} \mathsf{E}(z_n - \overline{z})^2 = (1/N_l)\nu^2 \tag{3.6}$$

The most widely used method to obtain N_l independent realizations of a single pixel is to generate *multi-look images*. Let the time instant t_n, $n = 0, \cdots, N_l$, be defined as

$$t_n = -\tfrac{1}{2}T_a + \frac{n}{N_l}T_a \tag{3.7}$$

Then the aperture $(-\frac{1}{2}T_a, \frac{1}{2}T_a)$ is subdivided into N_l subapertures (t_{n-1}, t_n), $n = 1, \cdots, N_l$. The subaperture time is then T_a/N_l. Azimuth compression with each subaperture results in N_l independent images which are added incoherently. The resolution is then determined by the length of a single subaperture, and thus decreased by a factor N_l compared to the single look image. However, from equation (2.11) we know that subdividing the aperture in N_l is equivalent to subdividing the Doppler band into N_l subbands. Hence, the multiple looks can be generated by azimuth compression with the full aperture (and thus full Doppler bandwidth) and filtering with a bandpass filter to subdivide the Doppler bandwidth in N_l subbands. Let the frequency f_n, $n = 0, \cdots, N_l$, be defined as

$$f_n = \frac{2v^2}{\lambda R_0}\left(\tfrac{1}{2}T_a - \frac{n}{N_l}T_a\right) \tag{3.8}$$

Then the n^{th} bandpass filter has bandwidth $B_n = 2v^2T_a/\lambda R_0 N_l$ and is defined over the frequency band (f_{n-1}, f_n). Both principles are illustrated in figure 3.3. The complexity of azimuth

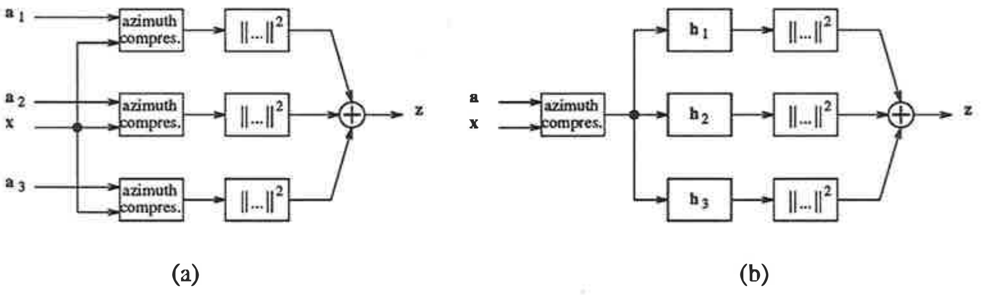


Figure 3.3: Multiple look azimuth compression with $N_l = 3$ (a) and full-resolution single look azimuth compression with $N_l = 3$ Doppler bandpass filters (b). The bandwidth of the bandpass filters is equal to the bandwidth of the subapertures.

compression is mainly determined by the length of the data signal. The complexity of both multiple look azimuth compression approaches are roughly determined by the azimuth compression. The bandpass filters in figure 3.3.b are in general short, thus the filtering complexity can be assumed negligible. Since the approach shown in figure 3.3.a requires N_l times an azimuth compression, its complexity will be N_l times larger than the complexity of the approach shown in figure 3.3.b.

Geometric correction

The main geometric distortion in a SAR image is caused by the non-linear relationship between the temporal variable in range t' and the spatial ground variable y. A side effect of the non-linear relationship between t' and y is that the range resolution mapped on the ground, say δy, will vary within the swath. Let $\phi = \arccos(h/R)$, then this increase is due to the relationship $\delta y = \delta r / \sin\phi$, see figure 3.4. δy would be worse than δr for $\phi < \pi/2$. The mapping of the SAR image onto a rectangular grid on the ground plane is also referred to as *slant-to-ground* conversion.

3.3 Real-Time SAR Processing for PHARUS

In this section we derive the system parameters of a real-time on-board SAR processor for the PHARUS system [HSKP92]. The PHARUS system is a fully polarimeteric C-band airborne SAR. It has an active antenna array that consist of 48 Transmit/receive (T/R) modules. Each T/R module has its own digital control unit that controls the phase and amplitude. It supports the internal calibration of the antenna, in that, the T/R modules can be interchanged without affecting the antenna beam. The antenna array has three clusters, each containing sixteen T/R

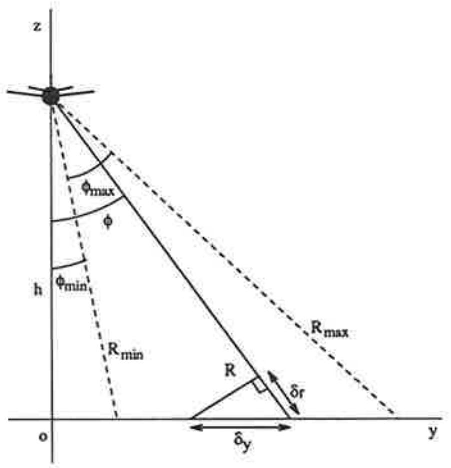


Figure 3.4: Relationship between slant range resolution δr and ground range resolution δy.

modules. Each cluster control calculates the individual settings per T/R module, depending on the position of the module within the array, the required tapering, required beam steering, etc.. The antenna is modular, in that, clusters can be added.

An important property of the PHARUS antenna is that beam can be electronically steered. For example, the drift of the aircraft can be compensated by yaw steering. But it can also be used for advanced SAR modes, such as *spotlight SAR* and *scan SAR*. Spotlight SAR increases the maximum aperture length by pointing the beam at a fixed spot on the ground, so that azimuth resolution can be increased. Scan SAR illuminates multiple strips on the ground, so that the maximum swath width increases.

The PHARUS is fully polarimetric, which means that each transmitted pulse can have a horizontal polarization or a vertical polarization. The antenna receives both horizontal and vertical polarized pulse echoes. In general, the polarization will switch from pulse to pulse, thus we obtain four SAR echo data sets, usually denoted as HH, HV, VH, and VV (the first capital stands for the polarization of the transmitted pulse, the second capital stands for the polarization of the antenna during the recepetion of the pulse echo). The four data sets results in four geometrically identical images. However, objects and areas within the image may differ in their radiometric appearances, which is important for forest and agriculture classification purposes and contrast enhancement of images.

SAR processor specifications

We assume that for on-board processing purposes it is sufficient to process one channel (in general this is either the VV or HH data sets). The SAR processing specifications, see table 3.1,

SAR parameters		
wavelength	λ	0.057 m
pulse bandwidth	B_p	45 MHz
Doppler band	B_d	146 Hz
squint angle	$\|\gamma\|$	< 3°
pulse width	τ_p	12.8 μsec
minimal range	R_{min}	7 km
maximal range	R_{max}	16 km
nominal speed	v	150 m/sec
range sampling frequency	f_r	50 MHz
azimuth sampling frequency	f_a	625 Hz

Table 3.1: PHARUS system parameters.

are based on the PHARUS high-resolution non-polarimetric mode, specified in [Ott94]. In this mode the highest input data rate of the SAR processor is expected. For this reason we consider the figures corresponding to this mode representative for the feasibility of a real-time SAR processor. In this application we omit the slant-to-ground conversion. It will be addressed in the next section, and can be easily applied here.

The length of the pulse replica is $N_p = 640$. Let $W_r = R_{min} - R_{max} = 9$ km be the swath width in range, and let $\delta r = \frac{1}{2}c/f_r = 3$ m be the sample distance in range. Then the number of pulse echo samples per pulse echo is $N_r = N_p + W_r/\delta r = 3640$. The number of samples that is written to the corner turning memory is $N_r - N_p = 3000$. The effective data rate of the range compression is the number of samples that has to be processed within one azimuth sampling period $\mathrm{D}_r = N_r f_a = 2.3$ MHz.

From equations (2.14) and (2.17) follows that the optimum range and azimuth resolutions of PHARUS are $\rho_r = 3.0$ m and $\rho_a = 1.0$ m, respectively. However, it is desirable to have square resolution cells, thus requiring an azimuth resolution $\delta x = 3$ m. Let B_a be the Doppler bandwidth that corresponds to δx, then $B_a = v/\delta x = 49$ Hz. Thus we can construct $N_l = \lfloor B_d/B_a \rfloor = 3$ non-overlapping subapertures to generate multiple look images. However, a rule of thumb is that up to 50 % overlap between adjacent subapertures is allowed, thus we can increase the number of looks to $N_l = 5$. We propose the multiple look generation scheme shown in figure 3.3.b.

One might conclude that decimation of the azimuth sampling frequency is allowed from the fact that the azimuth sample frequency is sufficiently larger than the Doppler bandwidth. However, noise energy from outside the Doppler band may appear inside the Doppler band, due to aliasing. We can overcome this problem by prefiltering with a bandpass filter, but then we meet the following problem. The PHARUS system has a range dependent squint angle γ. Yaw steering of the antenna beam only yields zero squint for a specific range. For smaller or large range the yaw steering yields a non-zero squint angle. The squint angle induces a variable Doppler centroid $|f_{dc}| < 275$ Hz. The bandpass filter should thus have a variable

SAR processing parameters		
range compression data rate	D_r	2.3 Msamples/sec
number of pulse echo samples	N_r	3000
number of pulse samples	N_p	640
number of looks	N_l	5
data reduction in azimuth	$n_\downarrow$	4
azimuth compression data rate	D_a	10 Msamples/sec
number of samples per azimuth batch	N_b	940
depth of focus	N_{dof}	23
max. size 2D azimuth filter in range	N_{mig}	8
max. size 2D azimuth filter in azimuth	N_a	470
latency	$\mathcal{L}$	9 sec

Table 3.2: Real-time SAR processor parameters for PHARUS.

center frequency. After filtering, we can decimate the azimuth echo signal with a decimation factor $n_\downarrow = \lfloor f_a/B_d \rfloor = 4$.

For the worst-case situation ($|\gamma| = 3°$ and maximum range) range migration must be compensated. From equation (2.20), we have the aperture time for maximum range is $T_a = 3$ sec. Thus, using equation (2.28), we have the maximum decline ΔR_{mig} of $R(t)$ over one aperture $\Delta R_{mig} = \Delta R(\frac{1}{2}T_a) - \Delta R(-\frac{1}{2}T_a) = vT_a \sin|\gamma| = 23$ m. The size of the 2D azimuth filter that corresponds to maximum range is then $N_a = T_a f_a / n_\downarrow = 470$ in azimuth and $N_{mig} = 8$ in range. Furthermore, from equation (2.26), we have the depth of focus $\Delta R_{dof} = 70$ m, and thus $N_{dof} = 23$.

The overlap that we have to take into account in designing the corner turning memory is the size of the 2D azimuth filter in azimuth *before* data reduction $n_\downarrow N_a$. Let N_b be the azimuth batch size *after* data reduction. A trade-off between latency and utilization leads to an utilization $U = 50$ %, and thus $N_b = 940$, and a latency $\mathcal{L} = n_\downarrow N_b(1 + U)/f_a = 9$ sec. In this case the data rate for azimuth compression after the data reduction is $\mathsf{D}_a = N_{mig}(N_r - N_p)f_a/n_\downarrow U = 10$ Msamples/sec.

The specifications for a real-time SAR processor for PHARUS are summarized in table 3.2. One must realize, however, that designing a real-time SAR processor is more than the determination some performance figures. Perhaps the most bounding constraints in the design are the specification of the specific hardware components. Therefore, these figures should be interpreted as representative for the PHARUS system, which gives an impression of the dimensions of the parameters of a real-time SAR processor. In [Bie94a] a similar study is presented, based on preliminary specifications of PHARUS.

3.4 Real-Time SAR Processing for ERS-1

In satellite SAR huge amounts of data are involved, typically in the order of 100 Mbytes for imaging a 100×100 km surface. This figure will increase more and more in future SAR systems. The communication system and the intermediate storage must handle this increasing amount of data as well. Furthermore, ground stations must process, broadcast and archive the SAR data and therefore show the tendency to become more and more expensive and time-consuming due to this data increment.

An attractive solution to reduce the communication requirements is to perform SAR processing on-board the satellite. In combination with image compression techniques (e.g. JPEG), it is the most rigorous kind of raw SAR data reduction. For example, an ERS-1 SAR image contains 63 Mbytes whereas a batch of raw ERS-1 SAR data contains 300 Mbytes. Image compression techniques can be used to achieve an even higher data reduction ratio. The advantages of data reduction by using on-board SAR processing and image compression can be summarized as follows:

Intermediate data storage: As a result of the compression rates achieved, more data can be stored on-board and less ground stations will be required. Moreover, the SAR can collect data from parts of the earths surface which are not within the reach of a ground station at that moment.

Direct broadcast facility: Since the satellite transmits the end-product down to earth, the end-user has the SAR image directly available. This means that a processing and broadcasting facility at the ground stations are not required. The end-users also receive the most up-to-date information.

Fast archiving: The image received by a ground station can be monitored immediately and, as a consequence, can be archived immediately.

In this section we will derive the specifications of a real-time on-board satellite SAR processor based on ERS-1 specifications. The objective is to optimize the function specifications and the processing performance parameters given the requirements and the specified SAR parameters.

SAR processing specifications

The specifications of the real-time SAR processor are based on the ERS-1 product specifications [ESA92]. Typical SAR parameters are listed in table 3.3. Based on the defined parameters we can give the specifications of a real-time SAR system. For the on-board processing we will consider the so-called "fast delivery" product specifications:

• required ground range resolution:	$\delta y \leq 33$ m
• required azimuth resolution:	$\delta x = 33$ m

SAR parameters		
wavelength	λ	0.057 m
Doppler bandwidth	B_d	1300 Hz
pulse bandwidth	B_p	15.5 MHz
squint angle	γ	$\approx 0°$
pulse width	τ_p	37.1 μsec
minimum range	R_{min}	850 km
maximum range	R_{max}	890 km
nominal speed	v	7000 m/sec
nominal altitude	h	800 km
range sampling frequency	f_r	19 MHz
azimuth sampling frequency	f_a	1700 Hz

Table 3.3: ERS-1 system parameters.

• no. of looks:	$N_l = 3$
• ground range swath width:	$W_g = 100$ km

From equation (2.14) follows that the optimum range resolution after pulse compression is $\rho_r = 9.7$ m. Consider the angles $\phi_{min} = \arccos(h/R_{min})$ and $\phi_{max} = \arccos(h/R_{max})$, see figure 3.4. The ground resolution is then $\delta y = \rho_r/\sin\phi_{min} = 29$ m at minimum range and $\delta y = \rho_r/\sin\phi_{max} = 22$ m at maximum range, and thus meets the specification. After the range compression, the effective number of samples per pulse echo that must be written in the corner turning memory is $N_r - N_p = 5000$. The effective data rate required for the range compression is defined as the number of range samples that is processed within the inter-pulse time $\mathsf{D}_r = N_r f_a = 9.7$ Msamples/sec.

The length of the pulse replica is $N_p = 700$. Let $W_r = R_{min} - R_{max} = 40$ km be the swath width in range and let $dr = \frac{1}{2}c/f_r = 7.9$ m be the sample distance in range. Then the number of pulse echo samples is $N_r = N_p + W_r/dr = 5700$. Observe that the specified minimum and maximum ranges correspond to a ground swath width $W_y = 100$ km.

Recall equation (2.20), which gives the subaperture length that corresponds to a required azimuth resolution δx at range R_0. We include an increment factor c_0 that allows azimuth filter weighting to increase the PSLR, without affecting the required resolution

$$L_a = c_0 \frac{\lambda R_0}{2\delta x} \qquad (3.9)$$

We shall assume that $c_0 = 1.5$ (Hamming window). A three look image requires three successive apertures of length L_a. Let B_a be the equivalent Doppler bandwidth per aperture, then, from

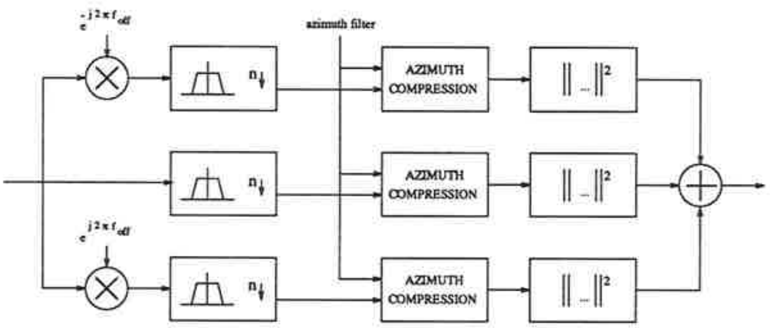


Figure 3.5: Multilook azimuth compression with frequency translation over $-f_{off}$, 0, and f_{off}, respectively, lowpass filtering, decimation with a factor $n_\downarrow$, azimuth compression and multiple look summation.

equation (2.11), follows

$$B_a = \frac{2vL_a}{\lambda R_0} = c_0\frac{v}{\delta x} \tag{3.10}$$

Since $B_a < B_d$, for each look the bandwidth (and thus the sample frequency) can be reduced. Substitute $\delta x = 33$ m, then the required Doppler bandwidth per look is $B_a = 320$ Hz.

The antenna beam of ERS-1 is yaw steered, yielding zero squint angle γ. Substituting δx, ρ_r and the system parameters in equation (3.1), leads to the conclusion that range migration compensation is not required. From equation (2.26) we have the depth of focus $\Delta R_{dof} = 76$ km. Observe that the depth of focus is larger than the swath width in range W_r, thus one azimuth filter is sufficient for azimuth compression of the complete data set. The subaperture length corresponding to range $\frac{1}{2}(R_{max} - R_{min}) = 870$ km is $L_a = 1.1$ km.

The critical azimuth sampling frequency required for the processing is determined by the required Doppler bandwidth per subaperture B_a. The decimation factor after proper band filtering is $n_\downarrow = \lfloor f_a/B_a \rfloor = 5$. However, in general, off-the-shelf decimation hardware components require powers of 2, and therefore we will set $n_\downarrow = 4$.

To reduce the data rate we split the azimuth echo signal in three channels and mix the three Doppler bands to baseband, see figure 3.5. We can now use low-pass filters with real taps, and then decimate the data rate with factor $n_\downarrow = 4$. An additional advantage is that the Doppler bands of the three channels are mixed to baseband, so we can use one azimuth filter for all three signals. Let $dx = v/f_a = 4.1$ m be the sample distance in azimuth, then the length of the decimated azimuth filter is $N_a = L_a/n_\downarrow dx = 67$. Directly after the azimuth compression, the samples are squared and summated.

Let N_b be the azimuth batch size *after* data reduction, then we specify (somewhat heuristically) an utilization $U = 90$ %, thus $N_b = 670$. The latency is $\mathcal{L} = n_\downarrow N_b(1 + U)/f_a = 2.3$

SAR processing parameters		
range compression data rate	D_r	9.7 Msamples/sec
number of pulse echo samples	N_r	5700
number of pulse samples	N_p	700
number of looks	N_l	3
data reduction in azimuth	$n_\downarrow$	4
azimuth compression data rate	D_a	8.5 Msamples/sec
number of samples per azimuth batch	N_b	275
size 1D azimuth filter in azimuth	N_a	67
number of image pixels in ground range	N_y	6100
latency	$\mathcal{L}$	2.3 sec

Table 3.4: ERS-1 SAR processor parameters.

sec. In this case the data rate for azimuth compression after the data reduction is $\mathsf{D}_a = N_l(N_r - N_p)f_a/n_\downarrow U = 8.5$ Msamples/sec.

It remains to do the slant-to-ground conversion. It is desirable to have square pixels, mapped on the xy-plane. The pixel size in azimuth is $n_\downarrow dx = 16.4$ m. Therefore the image must be interpolated[2] in range to obtain the required ground range pixel size $dy = 16.4$ m. One ground range image line in azimuth is constructed by interpolation multiple subsequent slant range image lines. Let $N_y = W_y/dy = 6100$ be the image size in ground range direction, thus after the interpolation.

The specifications for a real-time SAR processor for ERS-1 are summarized in table 3.4. These figures gives a good impression of the performance of a real-time on-board SAR processor for the ERS-1 satellite. In [BvHvB94] a demonstrator on-board processor based on these figures was presented.

3.5 Concluding Remarks

In this section we have specified the processing parameters for real-time on-board SAR processing for the airborne SAR system PHARUS and the satellite SAR system ERS-1. Obviously, data rates of azimuth and range compression remain the most challenging problem in designing real-time SAR processor hardware, see tables 3.2 and 3.4. Using dedicated or commercially available state-of-the-art DSP boards can perform the range and azimuth compression, but constraints in processor size and power consumption may not be met. Obviously, these constraints are tight, especially if the processing should be performed on-board a satellite.

Off-the-shelf components can be used to perform the bandpass filtering, interpolation, pythagoras processing. Implementation of the corner turning memory is straight-forward,

[2]For example, cubic spline interpolation.

depends mainly on the commercially available memory components. Most effort should be in designing of dedicated range and azimuth compression hardware. For example, range and azimuth compression with an effective data rate of 10 Msamples/sec implemented in commercially available state-of-the-art DSP boards [Cat92], will have $\mathcal{O}(100)$ W power consumption. The size will be approximately equivalent to six Euro-6 boards, but this may be too optimistic. In this estimation we did not take into account I/O data handling.

With this in mind, we focus in the next three chapters on the design of efficient convolution hardware architectures. We do not follow the conventional approach in designing the convolution hardware. This will not lead to satisfying reduction of processor size and power consumption. Our approach starts with an extensive analysis of the convolution problems, and leads to a *generic* solution for long convolutions running on extremely high effective data rates. It will reduce the amount of hardware required for range and azimuth compression to a few dedicated chips.

CHAPTER 4

CONVOLUTION AND MULTIRATE SIGNAL PROCESSING

4.1 Introduction

In this chapter we will concentrate on one of the critical steps in SAR processing, namely the convolution required for the range and azimuth compression.[1] Although much is written on fast convolution algorithms (see e.g. [Bla85]), in SAR processing the required convolution length in combination with the data rate remains a challenging complexity.

As we have mentioned in the previous chapter, in SAR processing signals can have lengths of several Ksamples. In the future, this number will increase to more than 10 Ksamples. The reference signal (the pulse replica or the azimuth filter, as described in the previous chapters) can have lengths up to 1 Ksamples. For range compression, processing time may not exceed the inter pulse time, which is in the order of mseconds. Performing the convolution in the time domain would then require an effective data rate of the processor of several GFlops. Performing the convolution in the frequency domain would still require an effective data rate of several hundreds of MFlops. The problem becomes even tougher if the processor size must be minimized.

In this chapter we introduce novel techniques that will lead to fast convolution algorithms and architectures. To this end we rely on multirate signal processing. Multirate signal processing was originally applied in telecommunications [BD74]. Since then, research effort has increased, especially in designing aliasing free multirate filter banks, see e.g. [CR83, Vet87, Vai90]. In the late 80's the research effort moved to applications, such as FIR filtering with multirate filter banks [Vet88], filter bank convolvers [Vai93, PV95] and adaptive filtering [GV92].

[1] Observe that the structure of digital correlation is almost equivalent to the structure of digital convolution.

At this moment however, the multirate signal processing theory lacks application-specific views, i.e. it is not yet common to use the multirate signal processing theory as a tool to solve the numerous digital signal processing problems, although some attempts have been made as we have said above. Generally, multirate signal processing theory focuses on multirate filter bank systems. However, long convolution with high data rates is a difficult problem, and multirate signal processing is a way to reduce complexity substantially.

Thus we investigate the usage of multirate signal processing in relation with long convolutions. The successive steps will lead to a description of an optimized *multirate convolution system*. An approach based on algebraic manipulations that led to similar convolution systems, referred to as *perfect convolution filter banks*, was presented in [Ste91]. The methodology that we propose, however, is akin to the method of combined algorithm development and architecture design as described in [Dep93] and the method of engineering algorithms as proposed in [McW92, MP92]. This methodology allows direct mapping of an algebraic specification of signal processing algorithms - here a multirate convolution - into prototyping architectures or dedicated VLSI processors. The mapping is obtained through graphical manipulations rather than by manipulating multi-indexed formulas.

In this chapter we derive the fundamentals of our methodology. We present the relationship between the graphical manipulations of the algorithm and its equivalent algebraic representations from multirate signal processing theory. It is known that multirate convolution is close to *block-signal processing*. In fact, if we embed the Discrete Fourier Transform (DFT) in the multirate convolution system hierarchically, we can show that it is equivalent to overlap-add or overlap-discard convolution [OS89]. This point of view is taken in the next chapter, where we derive a convolution architecture based on block-signal processing. Both chapters end up with two descriptions of the optimized multirate convolution system, the first derived form a multirate signal processing point of view, the second from a block-signal processing point of view.

The outline of this chapter is as follows. First we give the preliminaries in section 4.2. In section 4.3 we introduce the multirate convolution system from an algorithmic view point rather than from the traditional filter bank view point. The concepts are extended in section 4.4, where the DFT becomes an important aid to speed up computations.

4.2 Notations and Preliminaries

In this section we give some notations and preliminaries of multirate signal processing. It is not the intention to give an overview of multirate signal processing. Rather this section serves as a basis for the following sections in this chapter.

Consider the M-channel multirate system of figure 4.1. Each m^{th} channel contains subsequently an advance z^m, an N-fold decimator, a constant multiplication factor d_m, an N-fold expander and a delay z^{-m}. The signal on input is either denoted as a vector $\mathbf{x}$ or a series $X(z)$. The signal on output is denoted as $\mathbf{y}$ or $Y(z)$.

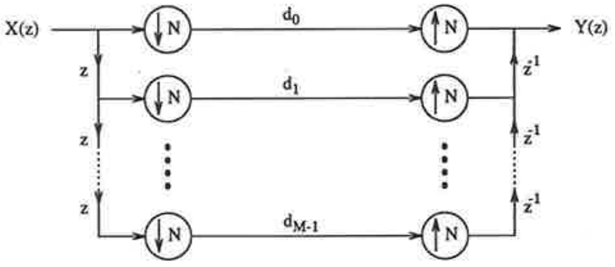


Figure 4.1: Basic M-channel multirate system with N-fold decimators/expanders and channel multiplication factors d_m, for $m = 0, \cdots, M-1$. By convention we use advances (z) on input and delays (z_{-1}) on output.

The input of the N-fold decimation in the m^{th} branch is $z^m X(z)$. The output of the N-fold decimation is denoted as $[z^m X(z)]_{\downarrow N}$. A matrix description of the successive steps (delay and N-fold decimation) on the m^{th} branch is illustrative. Let $\mathbf{Z}$ be the unitary shift operator

$$\mathbf{Z} = \begin{bmatrix} \ddots & & & & \\ & 0 & 0 & 0 & \\ & 1 & \boxed{0} & 0 & \\ & & 1 & 0 & \\ & & & & \ddots \end{bmatrix}$$

The box denotes the $(0,0)^{th}$ entry and serves as a reference of center. Then $\mathbf{Z}^m\mathbf{x}$ is the vector notation of $z^m X(z)$

$$[\cdots\ x_{m-1}\ \boxed{x_m}\ x_{m+1}\ \cdots] = [\cdots\ x_{-1}\ \boxed{x_0}\ x_1\ \cdots]\,\mathbf{Z}^m$$

The box denotes the zeroth entry. Let $\mathbf{N}_\downarrow$ be the N-fold decimation operator

$$\mathbf{N}_\downarrow = \begin{bmatrix} \ddots & & \\ & 1 & \\ & 0 & \\ & \vdots & \\ & 0 & \\ & & \boxed{1} \\ & & 0 \\ & & \vdots \\ & & 0 \\ & & & \ddots \end{bmatrix} \begin{matrix} \\ \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} N-1 \textit{ zeros} \\ \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} N-1 \textit{ zeros} \\ \\ \end{matrix}$$

Then the vector notation of the N-fold decimation $[z^m X(z)]_{\downarrow N}$ is

$$[\cdots\ x_{m-N}\ \boxed{x_m}\ x_{m+N}\ \cdots] = [\cdots\ x_{-1}\ \boxed{x_0}\ x_1\ \cdots]\mathbf{Z}^m\mathbf{N}_\downarrow \tag{4.1}$$

Similarly, $\mathbf{N}_\uparrow$ is the N-fold expander: $\mathbf{N}_\uparrow = \mathbf{N}_\downarrow^t$. The output of the N-fold expander on the m^{th} branch, including the delay z^{-m}, is

$$[\cdots\ x_{m-N}\ \underbrace{0\ \cdots\ 0}_{N-1 \textit{ zeros}}\ \overset{\overset{m^{th}\ entry}{\downarrow}}{x_m}\ \underbrace{0\ \cdots\ 0}_{N-1 \textit{ zeros}}\ x_{m+N}\ \cdots]\, d_m = \mathbf{Z}^m\mathbf{N}_\downarrow d_m \mathbf{N}_\uparrow \mathbf{Z}^{-m}$$

Finally, the vector form of $Y(z)$ is

$$[\cdots\ y_{-1}\ \boxed{y_0}\ y_1\ \cdots] = [\cdots\ x_{-1}\ \boxed{x_0}\ x_1\ \cdots]\sum_{m=0}^{M-1} \mathbf{Z}^m\mathbf{N}_\downarrow d_m \mathbf{N}_\uparrow \mathbf{Z}^{-m}$$

Let $\mathbf{T}$ be the system transfer matrix,

$$\mathbf{y} = \mathbf{x}\mathbf{T} \tag{4.2}$$

Then

$$\mathbf{T} = \sum_{m=0}^{M-1} \mathbf{Z}^m\mathbf{N}_\downarrow d_m \mathbf{N}_\uparrow \mathbf{Z}^{-m} \tag{4.3}$$

Observe that for all m

$$\mathbf{Z}^m\mathbf{N}_\downarrow\mathbf{N}_\uparrow\mathbf{Z}^{-m} = \operatorname{diag}\{\cdots,0,1,\underbrace{0,\cdots,0}_{N-1\ zeros},\overset{(m,m)^{th}\ entry}{\overset{\downarrow}{1}},\underbrace{0,\cdots,0}_{N-1\ zeros},1,0,\cdots\}$$

Hence, $\mathbf{T}$ is a diagonal matrix

$$\mathbf{T} = \operatorname{diag}\{\cdots,t_0,\cdots,t_{N-1},\boxed{t_0},\cdots,t_{N-1},t_0,\cdots,t_{N-1},\cdots\}$$

where

$$t_n = \sum_{k=0}^{K_n-1} d_{n+kN}, \qquad K_n = \left\lceil \frac{M-n+1}{N} \right\rceil \tag{4.4}$$

Now we can state the following properties of multirate systems restricted to this special case:

- If $N \leq M$ and if the d_m are chosen such that $\mathbf{T} = \mathbf{I}$, then the system has the so-called perfect reconstruction property, i.e. $Y(z) = X(z)$.
- If $N = M$ then the system is said to be maximally decimated. If a maximally decimated system also is perfect reconstructing, then $d_m = 1$ for all m.
- If $N > M$, moreover, then $K_n = 0$ (and thus $t_n = 0$) for $n = M,\cdots,N-1$, and consequently $Y(z) \neq X(z)$.

Observe that if the multirate system is maximally decimated then the left-hand side of the system (the advance/channel splitting and the decimation) and the the right-hand side (the delay/channel combining and the expansion) are in fact a serial-to-parallel conversion and a parallel-to-serial conversion, respectively.

4.3 Multirate Convolution Systems

In the previous section we have introduced a multirate system which splits a signal into M channel signals which has reduced data rate. Furthermore, conditions have been determined under which the original input signal can be reconstructed from the decimated M channel signals. As the channel signals have reduced data rates one could ask whether it is possible to transform a complex operation on the incoming signal into M smaller operations on the decimated channel signals. This can have many advantages in real-time digital signal processing, where data rates of incoming signals might be much higher than the maximum processing speed of hardware

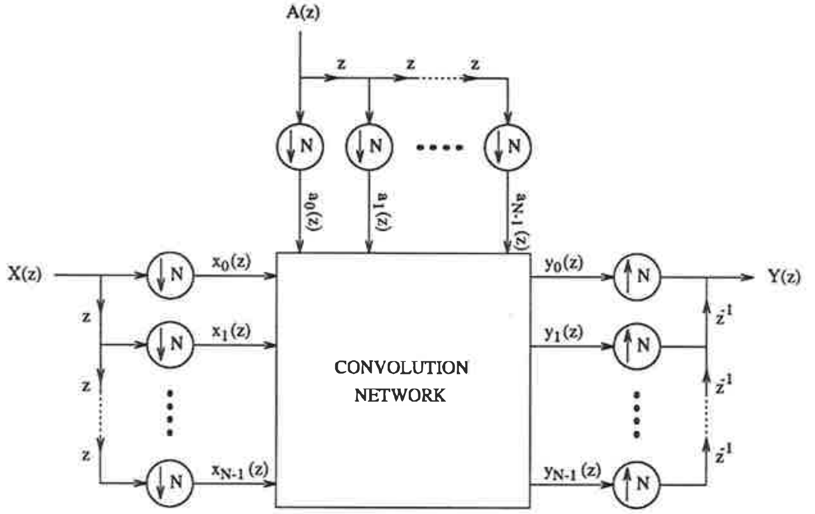


Figure 4.2: General form of a maximally decimated multirate convolution system. The convolution network performs an operation on the N incoming N-fold decimated channel signals $a_n(z)$ and $x_n(z)$ resulting in N output channel signals $y_n(z)$, such that $Y(z) = A(z)X(z)$.

components. Moreover, long data lengths can also be a potential problem since memory size is often limited. Both problems frequently occur in radar signal processing, as we have seen in chapter 3. The main operation is convolution of an incoming radar echo signal with a reference signal. We will consider the case that both the incoming echo signal, say $X(z)$, and the reference signal, say $A(z)$, are split up into M channel signals and are N-fold decimated. We will confine to the maximally decimated case, i.e. $M = N$. The objective is now to find a convolution network in which the channel signals are fed. The convolution network has an output of N channel signals, from which an output signal $Y(z)$ can be constructed such that $Y(z) = A(z)X(z)$. This multirate convolution system is shown in figure 4.2.

Our starting point will be the observation that the signals $A(z)$ and $X(z)$ can be written as

$$A(z) = \sum_{n=0}^{N-1} a_n(z^N)z^{-n}, \qquad X(z) = \sum_{n=0}^{N-1} x_n(z^N)z^{-n} \tag{4.5}$$

The N functions $a_n(z)$ and $x_n(z)$ are known as the *polyphase components* of order N of $A(z)$ and $X(z)$, respectively [CR83, Vet87, Vai90]. The n^{th} polyphase components are obtained as $x_n(z) = [z^n X(z)]_{\downarrow N}$ and $a_n(z) = [z^n A(z)]_{\downarrow N}$, which are the outputs of the n^{th} N-fold decimators in figure 4.2. Observe that the vector notation of the polyphase components are obtained in the previous section, see equation (4.1).

The linear convolution $Y(z) = A(z)X(z)$, in terms of the polyphase components of $A(z)$ and $X(z)$, is

$$Y(z) = \left(\sum_{n=0}^{N-1} a_n(z^N)z^{-n}\right)\left(\sum_{n=0}^{N-1} x_n(z^N)z^{-n}\right) \tag{4.6}$$

Observe that equation (4.6) looks very similar to a multiplication of two polynomials of order $N-1$. The only difference is that instead of having scalar coefficients, these polynomials have polyphase coefficients $a_n(z^N)$ and $x_n(z^N)$. Therefore it is convenient to introduce the following convention. Let $H(z) = \sum_{k=0}^{N-1} h_k(z^N)z^{-k}$ be an arbitrary polynomial in z in polyphase notation. Introduce the variable $w = z^N$. Then we can express $H(z)$ equivalently as an order $N-1$ polynomial in z with coefficients $h_n(w)$

$$H(z) = \sum_{n=0}^{N-1} h_n(w)z^{-n} \tag{4.7}$$

In order to distinguish the two notations we will write $H(z;w)$ if we mean equation (4.7). Observe that z^{-1} represents a unit-delay and thus w^{-1} represents the delay z^{-N}.

Equation (4.7) looks very similar to an $N-1$ order polynomial, with only difference that its coefficients are series in w. Applying the variable substitution to both sides of equation (4.6) gives

$$Y(z;w) = A(z;w)X(z;w) \triangleq \left(\sum_{n=0}^{N-1} a_n(w)z^{-n}\right)\left(\sum_{n=0}^{N-1} x_n(w)z^{-n}\right) \tag{4.8}$$

$A(z;w)$ and $X(z;w)$ are in fact as polynomials of order $N-1$ in z, thus their product, see equation (4.8), must be a polynomial of order $2N-2$ in z. But by convention $Y(z;w) \triangleq \sum_{n=0}^{N-1} y_n(w)z^{-n}$ is of order $N-1$ which implies that this interpretation is not valid. Nevertheless, we can interpret $A(z;w)$ and $X(z;w)$ as order $N-1$ polynomials and still satisfy equation (4.8), as we will show.

Define an intermediate polynomial $\widehat{Y}(z;w)$ of order $2N-2$ as

$$\widehat{Y}(z;w) = \sum_{n=0}^{2N-2} \widehat{y}_n(w)z^{-n} \tag{4.9}$$

with coefficients

$$\widehat{y}_m(w) = \sum_{n=0}^{N-1} a_n(w)x_{m-n}(w) \tag{4.10}$$

Observe that these coefficients are the result of a linear convolution of the N coefficients $\{a_n(w)\}$ and $\{x_n(w)\}$ of $A(z;w)$ and $X(z;w)$, respectively, and thus $A(z;w)X(z;w) \equiv \widehat{Y}(z;w)$. We can expand $\widehat{Y}(z;w)$ as

$$\begin{aligned}\widehat{Y}(z;w) &= \sum_{m=0}^{2N-2} \widehat{y}_m(w)z^{-m} \\ &= \sum_{m=0}^{N-1} \widehat{y}_m(w)z^{-m} + z^{-N}\sum_{m=0}^{N-2} \widehat{y}_{m+N}(w)z^{-m} \end{aligned} \tag{4.11}$$

Again let $z^N \leftarrow w$ then we have

$$\begin{array}{ll} \text{if } n = 0,\cdots,N-2 & \widehat{y}_n(w) + w^{-1}\widehat{y}_{n+N}(w) = y_n(w) \\ \text{if } n = N-1 & \widehat{y}_n(w) = y_n(w) \end{array} \tag{4.12}$$

Hence $\widehat{Y}(z;w) \equiv Y(z;w)$ and thus equation (4.11) can be written as

$$A(z;w)X(z;w) = \sum_{n=0}^{N-1} y_n(w)z^{-n} \equiv Y(z;w) \tag{4.13}$$

What we have shown is that $Y(z;w)$ is equivalent to $A(z;w)X(z;w)$ which was shown by introducing the intermediate order $2N-2$ polynomial $\widehat{Y}(z;w)$. In practice, $y_n(w)$, $n = 0,\cdots,N-1$, are obtained by first calculating $\widehat{y}_n(w)$, $n = 0,\cdots,2N-2$, by performing the linear convolution operation of equation (4.10). The fact that the polynomials are polynomials in w does not corrupt the properties the linear convolution. Secondly, $y_n(w)$ $n = 0,\cdots,N-1$ is calculated directly by applying equation (4.12).

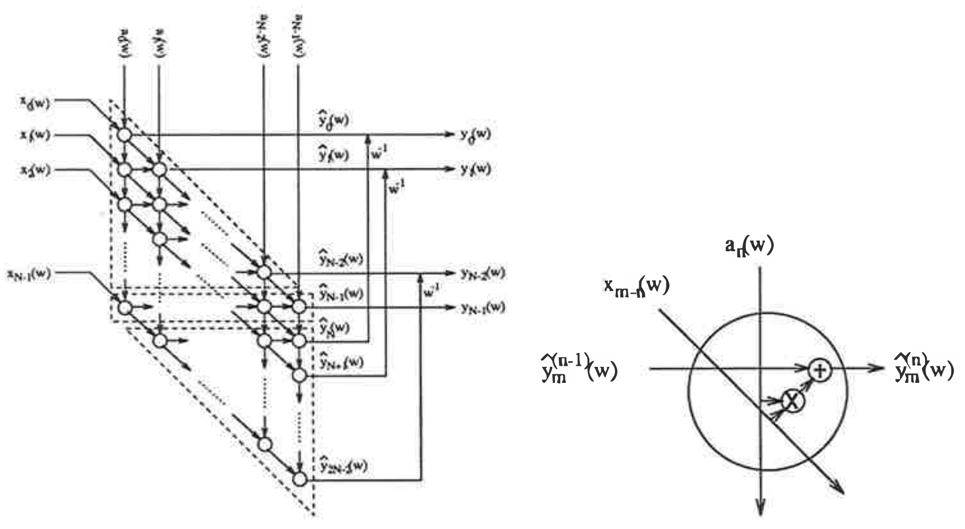
The linear convolution network

We can now derive a convolution network to be used in figure 4.2. To this end we represent equation (4.10) recursively [Kun88]

$$\widehat{y}_m^{(n)}(w) = \widehat{y}_m^{(n-1)}(w) + a_n(w)x_{m-n}(w), \quad \left\{ \begin{array}{l} \text{for } n = 0,\cdots,m, \text{ if } m = 0,\cdots,N-1 \\ \text{for } n = m-N+1,\cdots,N-1, \\ \qquad\qquad \text{if } m = N,\cdots,2N-2 \end{array} \right. \tag{4.14}$$

where $\widehat{y}^{(m)}(w)$ is the result of the m^{th} recursive step. The recursive representation is useful when an algorithm is highly regular (which is the case for convolution). In that case, data dependencies and regularities can easily be recognized.

The recursion has a *computational flow graph* or *dependence graph* representation [Kun88] shown in figure 4.3.a, where each *node* performs a multiply-add operation of polynomials in w, shown in figure 4.3.b. For convenience, the branch within the dependence graph that belongs to a zero input has been omitted. Usually the data samples involved in dependence graphs are scalars. In our case, however, the data samples are polynomials in w.



(a) (b)

Figure 4.3: Linear convolution network (a) and a node (b). The network performs the linear convolution of the sequences $\{a_n(w)\}$ and $\{x_n(w)\}$ and constructs the output sequence $\{y_n(w)\}$ via the intermediate sequence $\{\hat{y}_n(w)\}$.

The next step is to include the operation of equation (4.10) that maps the $2N-1$ outputs of the dependence graph to the N outputs of the convolution network. In practice this means that the $N-1$ outputs of the dashed lower triangle in figure 4.3.a are multiplied by w^{-1} and added to the $N-2$ outputs of the dashed upper triangle. The complete dependence graph is a typical realization of a convolution network that has input $x_n(w)$ and $a_n(w)$ and output $y_n(w)$, with $n = 0, \cdots, N-1$. Since we have used the linear convolution to derive this network we will refer to it as the linear convolution network.

The data flows within the linear convolution network only involve operations on polynomials in w or equivalently, using the fact that $z^N \leftarrow w$, operations in z^N. In practice this means that we can run the complete linear convolution network with a factor N reduced data rate. Observe that the N-fold decimators at the input of the multirate convolution system take care of the data reduction before the polyphase components are fed into the convolution network. Hence the linear convolution network is indeed a realization of the convolution network that we were looking for, with inputs and outputs running at reduced data rates.

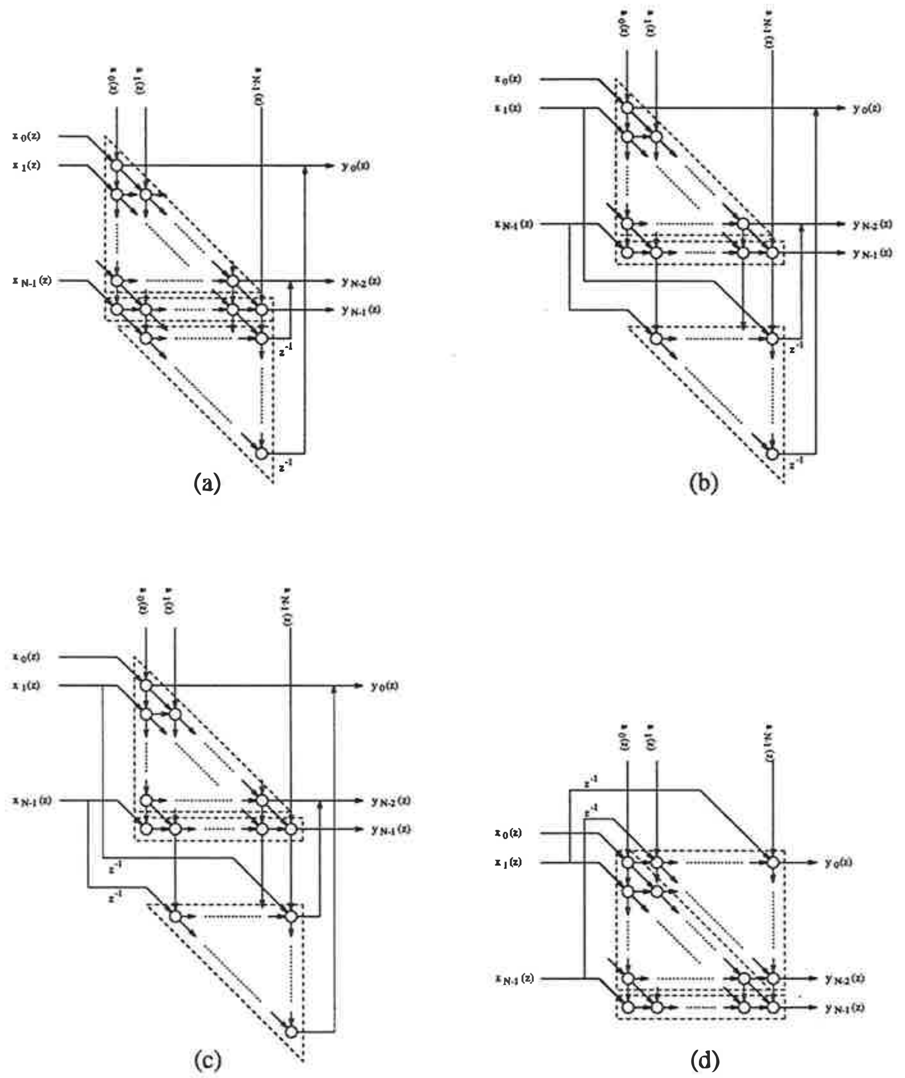


Figure 4.4: A typical example of algorithmic engineering: by manipulating of the linear convolution network (a) the circular convolution network (d) can be derived. The key rule is that dependencies of the intermediate convolution networks (b) and (c) are equivalent to the initial and resulting network (a) and (d), respectively.

The circular convolution network

Another realization of the convolution network can be obtained by using a circular convolution rather than a linear convolution. Instead of going through the algebraic derivations we derive this network graphically, with as starting point the linear convolution network of figure 4.4.a. The variable w has been replaced by z, i.e. implying that the network runs on a factor N reduced data rate. The basic fundamentals of the derivation of computing networks by manipulating of a dependence graph are given in [Kun88], and are worked out to a methodology in [McW92, MP92], referred to as *algorithmic engineering*. In the next chapter we will give more examples of the concept of algorithmic engineering.

The linear convolution network can be split-up in two parts given in figure 4.4.b. The lower triangle can be assumed as a linear operation on the $N-2$ input elements $x_n(z)$ followed by a delay z^{-1} on all $N-2$ output elements. This allows us to relocate the delay operators to all input branches. Now the $N-2$ output elements of the lower triangle are added to the $N-2$ output elements of the upper triangle, which is depicted in figure 4.4.c. Recall that the n^{th} outputs of both triangles are achieved recursively. Let the recursions be of the form

upper triangle: $\widehat{y}^{(n)}_{up,m}(z) = \widehat{y}^{(n-1)}_{up,m}(z) + c^{(n)}_{up,m}(z)$, for $n = 0, \cdots, m$

lower triangle: $\widehat{y}^{(n)}_{low,m}(z) = \widehat{y}^{(n-1)}_{low,m}(z) + c^{(n)}_{low,m}(z)$, for $n = m+1, \cdots, 2N-2$

where $c^{(n)}_{up,m}(z)$ and $c^{(n)}_{low,m}(z)$ are some update polynomials. Obtaining the m^{th} polyphase component $y_m(z)$ of $Y(z)$ is then simply the addition of the results of the recursions $y_m(z) = \widehat{y}^{(m)}_{up,m}(z) + \widehat{y}^{(2N-2)}_{low,m}(z)$. However, it is easy to see that $y_m(z)$ can also be obtained by first recursively computing $\widehat{y}^{(m)}_{up,k}(z)$ and then recursively computing $\widehat{y}^{(2N-2)}_{low,m}(z)$ with initial value $\widehat{y}^{(m)}_{low,m}(z) = \widehat{y}^{(m)}_{up,m}(z)$. Graphically, this can be represented by superimposing the lower triangle onto the upper triangle, as is shown in figure 4.4.d. This network is another realization which can be used within the multirate convolution system.

As we shall now show, this realization partly performs a circular convolution [OS89]. Let be given the $2N-1$ polynomial elements $\widehat{x}_m(z)$, $m = 0, \cdots, 2N-2$ and the N polynomial coefficients $a_n(z)$, $n = 0, \cdots, N-1$. Let $\widehat{\widehat{y}}_m(z)$, $m = 0, \cdots, 2N-2$, be the result of their circular convolution

$$\widehat{\widehat{y}}_m(z) = \sum_{n=0}^{N-1} a_n(z)\widehat{x}_{(m-n)\text{mod}(2N-1)}(z) \tag{4.15}$$

The recursive form of equation (4.15) is

$$\widehat{\widehat{y}}^{(n)}_m(z) = \widehat{\widehat{y}}^{(n-1)}_m(z) + a_n(z)\widehat{x}_{(m-n)\text{mod}(2N-1)}(z) \tag{4.16}$$

The dependence graph of equation (4.16) is shown in figure 4.5.a. Obviously, if we are only interested in the polynomial results $\widehat{\widehat{y}}_m(z)$, for $m = N, \cdots, 2N-2$, then we only have to

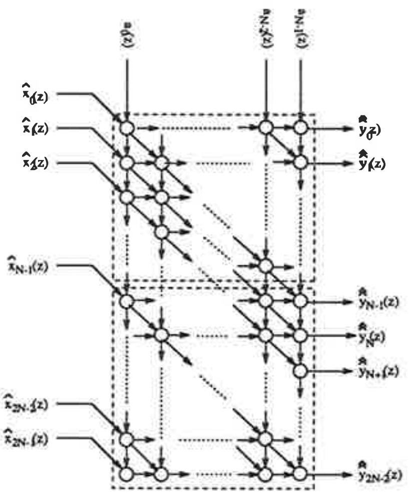
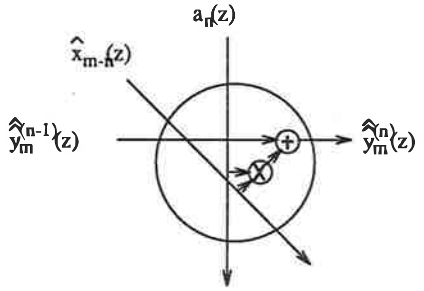


(a) (b)

Figure 4.5: Circular convolution network (a) and a node (b). The network performs the circular convolution of the sequences $\{a_n(w)\}$ and $\{\hat{x}_n(w)\}$ with as output result $\{\hat{\hat{y}}_n(w)\}$. Observe that the lower square of the circular convolution network has been derived from the linear convolution network, see figure 4.4.

compute the lower dashed rectangle. This part of the dependence graph is equivalent to the dependence graph of figure 4.4.d, if we let

$$\begin{array}{rcll} \hat{x}_m(z) & \leftarrow & \begin{cases} w^{-1}x_{m+1}(z) & \text{for } m = 0,\cdots,N-2 \\ x_{m-N+1}(z) & \text{for } m = N-1,\cdots,2N-1 \end{cases} \\ \hat{\hat{y}}_m(z) & \leftarrow & \begin{cases} \text{undetermined} & \text{for } m = 0,\cdots,N-2 \\ y_{m-N+1}(z) & \text{for } m = N-1,\cdots,2N-1 \end{cases} \\ a_n(z) & \leftarrow & a_n(z) \qquad\qquad\quad \text{for } n = 0,\cdots,N-1 \end{array} \tag{4.17}$$

Observe that the circular convolution network has the same complexity in terms of polynomial multiply-add operations. However, one advantage might be that it does not require delay-add operations at the output.

4.4 Multirate Convolution and the DFT

In this section we employ DFT techniques to speed up the multirate convolution system. It is well known that discrete convolution can be performed efficiently in the frequency domain.

Moreover, using the Fast Fourier Transform (FFT) to compute the DFT will even decrease the computational complexity of the convolution network in terms of polynomial multiplications. Furthermore we show that the multirate convolution system can be considered as a multirate system with DFT filter banks. The latter was first observed by [Por80] for the case that $A(z)$ is of order $N-1$, and in [Vet88] it was shown that the system works for $A(z)$ of any order.

In [Por80, Vet88] however, this observation has been made from the viewpoint of a multirate system with DFT analysis/synthesis filter banks and additional channel filters. It was stated that under certain conditions the multirate system can be considered as an overlap-add/overlap-discard convolution system based on short-time Fourier transforms.[2] We have shown in the previous section that the relation between the multirate systems and the overlap-add/overlap-discard convolution method can be explained without using the short-time Fourier transform and without reference to any kind of filter bank. In fact, we will show in this section that introducing the DFT within the multirate convolution system that we have proposed is nothing more than a fast convolution of two polynomial sequences. Moreover, other transforms could be used as well, for example, Mersenne or Fermat number transforms [Rad72, AB75, AC77].

The DFT for polynomial sequences

Our starting point is the DFT for a polynomial vector. It is customary to write expressions related to the DFT in terms of W_N, defined as $w_N \triangleq \exp(-j2\pi/N)$. Given a polynomial vector $\mathbf{x}(z) = [x_0(z) \ \cdots \ x_{N-1}(z)]$, then its N-point DFT is also a polynomial sequence $\boldsymbol{X}(x) = [X_0(z) \ \cdots \ X_{N-1}(z)]$, with components

$$X_m(z) = \sum_{n=0}^{N-1} x_n(z) W_N^{mn} \tag{4.18}$$

Conversely, if $\boldsymbol{X}(z)$ is given then its N-point IDFT is $\mathbf{x}(z)$, with components

$$x_n(z) = \frac{1}{N} \sum_{m=0}^{N-1} X_m(z) W_N^{-mn} \tag{4.19}$$

The DFT and IDFT can also be expressed in matrix notation. Let $\mathbf{W}_N$ be the $N \times N$ DFT matrix

$$\mathbf{W}_N = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N & W_N^2 & \cdots & W_N^{(N-1)} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & W_N^{(N-1)} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} \tag{4.20}$$

[2]In [Vet88], though, the starting point is a general multirate system with analysis/synthesis banks without using the short-time Fourier transform. However, this general case is only briefly pointed out and not discussed explicitly.

Then the matrix forms of equations (4.18) and (4.19) are

$$\text{DFT:} \quad \boldsymbol{X}(z) = \mathbf{x}(z)\mathbf{W}_N \tag{4.21}$$

$$\text{IDFT:} \quad \mathbf{x}(z) = \frac{1}{N}\boldsymbol{X}(z)\mathbf{W}_N^* \tag{4.22}$$

Convolution networks and the DFT

Using the DFT matrix alternative DFT domain convolution networks can be obtained. Observe that the linear convolution network performs the linear convolution of two N-dimensional polynomial vectors $\mathbf{a}(z) = [a_0(z) \cdots a_{N-1}(z)]$ and $\mathbf{x}(z) = [x_0(z) \cdots x_{N-1}(z)]$. Let $\hat{\mathbf{y}}(z)$ be the $2N-1$-dimensional polynomial vector with components given in equation (4.10). Then $\hat{\mathbf{y}}(z)$ is the linear convolution $\hat{\mathbf{y}}(z) = \mathbf{a}(z) * \mathbf{x}(z)$. Let $\mathbf{y}(z) = [y_0(z) \cdots y_{N-1}(z)]$, then the relation between $\hat{\mathbf{y}}(z)$ and $\mathbf{y}(z)$ is

$$\mathbf{y}(z) = \hat{\mathbf{y}}(z)\left[\frac{\mathbf{I}_N}{z^{-1}\mathbf{I}_{N-1} \mid 0}\right] \tag{4.23}$$

Define the polynomial vectors $\mathbf{a}_0(z)$ and $\mathbf{x}_0(z)$ of length M, as

$$\mathbf{a}_0(z) = [a_0(z) \cdots a_{N-1}(z)\, 0 \cdots 0], \quad \text{and} \quad \mathbf{x}_0(z) = [x_0(z) \cdots x_{N-1}(z)\, 0 \cdots 0]$$

where $M = 2N-1$, and let their M-point DFTs be $\boldsymbol{A}_0(z) = \mathbf{a}_0(z)\mathbf{W}_M$ and $\boldsymbol{X}_0(z) = \mathbf{x}_0(z)\mathbf{W}_M$. Then from the Fourier theory we know that $\hat{\mathbf{y}}(z)$ can be obtained as follows

$$\hat{\mathbf{y}}(z) = \frac{1}{M}\big(\boldsymbol{A}_0(z) \otimes \boldsymbol{X}_0(z)\big)\mathbf{W}_M^* \tag{4.24}$$

where $\otimes$ denotes element-wise multiplication. With the observation that

$$\mathbf{x}_0(z) = \mathbf{x}(z)[\,\mathbf{I}_N \mid 0\,], \qquad \mathbf{a}_0(z) = \mathbf{a}(z)[\,\mathbf{I}_N \mid 0\,] \tag{4.25}$$

we can express equation (4.24) as

$$\mathbf{y}(z) = \frac{1}{M}\Big(\big(\mathbf{x}(z)[\,\mathbf{I}_N \mid 0\,]\mathbf{W}_M\big) \otimes \big(\mathbf{a}(z)[\,\mathbf{I}_N \mid 0\,]\mathbf{W}_M\big)\Big)\left[\frac{\mathbf{I}_N}{z^{-1}\mathbf{I}_{N-1} \mid 0}\right] \tag{4.26}$$

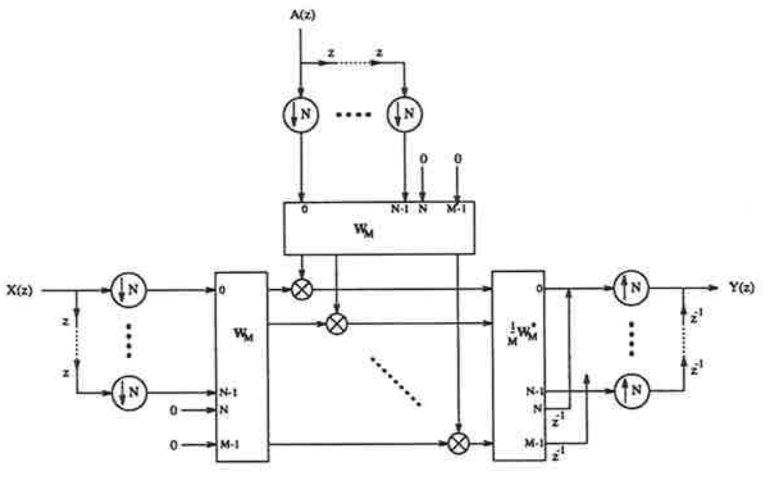
Equation (4.26) expresses the linear convolution property of the DFT for polynomial vectors.

A similar expression can be obtained for the circular convolution. Let be given

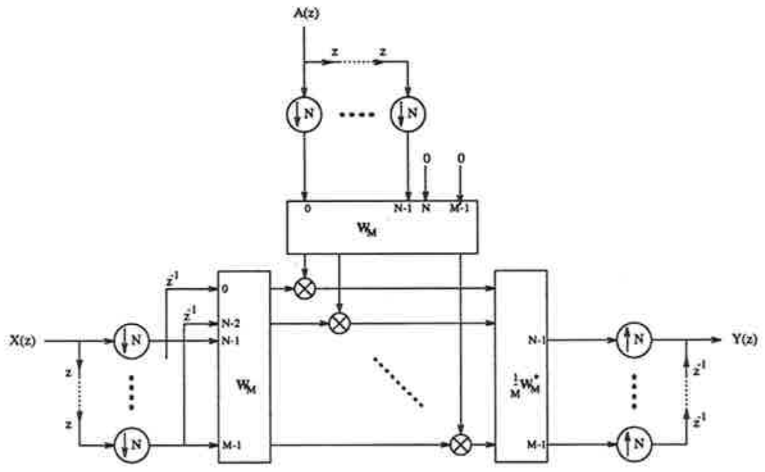
$$\hat{\mathbf{x}}(z) = [z^{-1}x_1(z) \cdots z^{-1}x_{N-1}(z)\;\; x_0(z) \cdots x_{N-1}(z)]$$

and denote $\widehat{\boldsymbol{X}}(z) = \mathbf{W}_M\hat{\mathbf{x}}(z)$ its M-point DFT. Let the circular convolution be $\hat{\mathbf{y}}(z) = \mathbf{a}(z) \circ \hat{\mathbf{x}}(z)$. Again, from the Fourier theory we know that $\hat{\hat{\mathbf{y}}}(z)$ can be obtained as follows

$$\hat{\hat{\mathbf{y}}}(z) = \frac{1}{M}\left(\boldsymbol{A}_0(z) \otimes \widehat{\boldsymbol{X}}(z)\right)\mathbf{W}_M^* \tag{4.27}$$



(a)



(b)

Figure 4.6: Alternative linear convolution network (a) and circular convolution network (b) using the DFT. The multiplications represents polynomial multiplications.

Define the the $N-1 \times N-1$ unitary shift matrix $\mathbf{Z}_{N-1}$ as

$$[\mathbf{Z}_{N-1}]_{i,j} = \begin{cases} 1 & \text{if } j = i-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.28}$$

then $\hat{\mathbf{x}}(z)$ can be obtained as

$$\hat{\mathbf{x}}(z) = \mathbf{x}(z) \left[\begin{array}{c|c} \dfrac{0}{z^{-1}\mathbf{Z}_{N-1}} & \mathbf{I}_N \end{array} \right] \tag{4.29}$$

Observe that

$$\mathbf{y}(z) = \hat{\hat{\mathbf{y}}}(z) \left[\dfrac{0}{\mathbf{I}_N} \right] \tag{4.30}$$

Now equation (4.30) can be expressed as

$$\mathbf{y}(z) = \frac{1}{M} \left(\left(\mathbf{a}(z)[\, 0 \mid \mathbf{I}_N \,]\mathbf{W}_M \right) \otimes \left(\mathbf{x}(z) \left[\begin{array}{c|c} \dfrac{0}{z^{-1}\mathbf{Z}_{N-1}} & \mathbf{I}_N \end{array} \right] \mathbf{W}_M \right) \right) \mathbf{W}_M^{*} \left[\dfrac{0}{\mathbf{I}_N} \right] \tag{4.31}$$

Equation (4.31) expresses the circular convolution property of the DFT for polynomial vectors.

Both equations (4.26) and (4.31) are algebraic representations of alternative convolution networks. They can be embedded in figure 4.2, such that we obtain the two multirate convolution systems of figures 4.6.a and 4.6.b. The structures of both multirate convolution systems based on the DFT have many advantages in terms of implementation. We not only have split up the long convolution operation into smaller linear or circular convolution operations but, if we use the FFT to compute the DFT, we have now also decreased the computational complexity in terms of polynomial multiplications.

Relationship to M-channel filter banks

The structures of figure 4.6 can be interpreted as M-channel multirate systems with polynomial DFT transformation and N-fold decimation/expansion, where $M = 2N-1$. These systems are extensively studied in e.g. [Vet86, SV86], where they are referred to as uniform DFT filter bank systems. The extension in our description is that the M channel signals are filtered by channel filters $\mathcal{A}_m(z)$. In fact, the channel filters are the M entries of the DFT vector $\mathcal{A}(z)$ of the vector $\mathbf{a}_0(z)$. This interpretation is illustrated in figure 4.7.

Figure 4.7 is equivalent to figure 4.6.a if we put

$$e_{0,m}(z) = \begin{cases} 1 & \text{for } m = 0, \cdots, N-1 \\ 0 & \text{for } m = N, \cdots, M-1 \end{cases}, \quad r_{0,m}(z) = 1 \quad \text{for } m = 0, \cdots, M-1$$

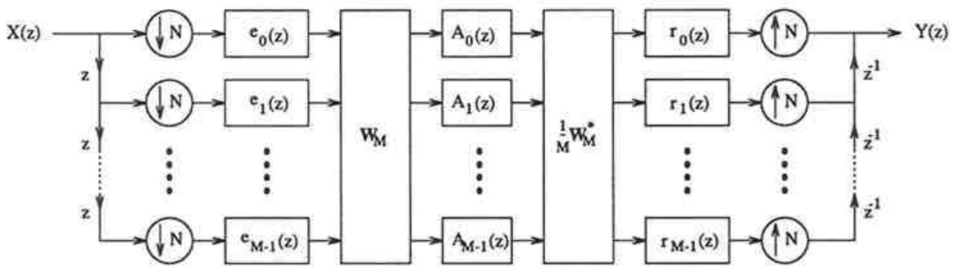


Figure 4.7: Filter bank representation of the multirate convolution systems based on the DFT. The channel filters $A_m(z)$ are the entries of the M-point DFT of the vector $\mathbf{a}(z)$.

and it is equivalent to figure 4.6.b if we put

$$e_{0,m}(z) = 1 \quad \text{for } m = 0, \cdots, M-1 \ , \qquad r_{0,m}(z) = \begin{cases} 0 & \text{for } m = 0, \cdots, N-2 \\ 1 & \text{for } m = N-1, \cdots, M-1 \end{cases}$$

In the latter case we take as input of the DFT the M-dimensional polynomial vector

$$\hat{\mathbf{x}}'(z) = [x_0(z) \ \cdots \ x_{N-1} \ z^{-1}x_0(z) \ \cdots \ z^{-1}x_{N-2}]$$

rather than $\hat{\mathbf{x}}(z)$, as was it assumed in figure 4.6.b. However, the only difference in the overall transfer function of the multirate convolution system is the delay $z^{-(N-1)}$.

4.5 Concluding Remarks

The use of the multirate convolution system instead of direct linear convolution has the advantage that it breaks up the long convolution in N^2 smaller convolutions (the polynomial multiplications within the nodes in the convolution network). It runs at a factor N reduced processing speed as a result of the N-fold decimators. The overall computational complexity is not reduced, but efficient architectures can be used which exploit the parallel structure of the convolution network and fast algorithms can be used for the small convolutions. Moreover, the convolutions within the nodes can be structured using the same multirate convolution system. This hierarchical approach can be repeated in principle, until the last convolution operation is nothing more than an atomic scalar multiplication (of course $X(z)$ and $A(z)$ must then be causal and of finite order).

The multirate convolution systems have well-known equivalents in digital signal processing theory. The linear and circular convolution networks can be interpreted as the overlap-add and overlap-discard convolution methods, respectively [OS89]. Both split the long linear convolution problem into multiple small convolution problems. The advantage of our multirate signal processing view is that, unlike the algebraic derivation, the convolution network shows a high degree of parallelization and regularity. In the next chapter we give formal relations between multirate signal processing and overlap-add and overlap-discard from a block-signal processing point of view. Again we will use graph manipulations to clarify the formalisms derived with algebra.

CHAPTER 5

CONVOLUTION AND BLOCK-SIGNAL PROCESSING

5.1 Introduction

In the previous chapter we have developed some algebraic methods based on multirate signal processing to solve the long linear convolution problem. In the literature it was often mentioned that multirate signal processing has lots of opportunities in efficient architecture design for convolution with high data rates and large convolution sequences [Por80, Vet88, Ste91]. However, these descriptions remains highly algebraic. They emphasize merely the theoretical behavior of such systems rather than the mapping of the proposed methods onto efficient algorithms and architectures. The point of view is now *block-signal processing*, which is traditionally the method to solve large convolution problems, see e.g. [AB74]. We start with the introduction of block-signal processing to solve the convolution problem, the so-called overlap-add and overlap-discard methods [OS89].

We use a formal methodology to come from the algebraic block-signal processing description to the efficient architectural realization. As mentioned in the previous chapter, this formal methodology is close to algorithmic engineering [McW92, MP92]. Algorithmic engineering is closely related to *systolic design* [Kun88]. The concept of systolic structures was originally seen as an architectural paradigm. However, systolic architectures as defined in [Kun88] are nothing more than structures of algorithms which could never provide realistic architectures. It is thus better to speak of systolic algorithms, and algorithmic engineering is a method to derive or transform such algorithmic structures. Once a systolic algorithm is developed the process of mapping it onto an array of parallel processors can be formalized. Examples of complex algorithms that can be handled in this way are matrix multiplication, QR- and SVDdecompositions, solving linear equations, and the like.

The key is to describe the algorithm graphically in a dependence graph, like the one in the previous chapter. The dependence graph can be determined from the recursive form of the algorithm and shows the dependencies between the operations within the algorithm. The dependence graph can then be projected onto a *signal flow graph.* Whereas in a dependence graph a *node* represents essentially a function (e.g. multiply-add), in a signal flow graph a node represents a *processor element.* This processor element is not necessarily a physical processor, it might be a communicating function on an abstract level. The advantage of a signal flow graph is that formal methodologies exist to map signal flow graphs into arrays of processors [Bu90, DHW93].

In the convolution networks the operations within the dependence graph were polynomial multiply-add operations. Here we have scalar multiply-add operations, as we show in section 5.2. In section 5.3 we map the dependence graph into a signal flow graph, in which we recognize the block structure of the overlap-add and overlap-discard methods. At the point that we introduce the short FFT in the signal flow graph, we may consider them equivalent to the linear convolution networks. In appendix D we already have shown that this holds from an algebraic point of view, and we have now derived the one-to-one relationship from an algorithmic point of view. In section 5.4 we exploit the properties of the short FFT to come to our final signal flow graph. It is in fact a highly regular and schematic description of the multirate convolution system based, presented in the previous section. The mapping onto an efficient hardware architecture is now straight-forward, as we will show in the next chapter.

5.2 The One-Dimensional Convolution Problem

Consider two finite scalar sequences $\mathbf{a} = [a_0 \cdots a_{N_a-1}]$ and $\mathbf{x} = [x_0 \cdots x_{N_x-1}]$, with $N_a \leq N_x$. Let the scalar sequence $\mathbf{y} = [y_0 \cdots y_{N_y-1}]$, with $N_y = N_a + N_x - 1$ be their linear convolution $\mathbf{y} = \mathbf{a} * \mathbf{x}$, then

$$y_n = \sum_{k=0}^{N_a-1} a_k x_{n-k} \tag{5.1}$$

In the previous section we have already mentioned that the linear convolution operation can be described recursively [Kun88]. The recursive form of equation (5.1) is

$$y_n^{(k)} = y_n^{(k-1)} + a_k x_{n-k}, \quad \begin{cases} \text{for } k = 0, \cdots, n, \text{ if } n = 0, \cdots, N_a - 1 \\ \text{for } k = 0, \cdots, N_a - 1, \text{if } n = N_a, \cdots, N_x - 1 \\ \text{for } k = n - N_x + 1, \cdots, N_a - 1, \text{if } n = N_x, \cdots, N_y - 1 \end{cases} \tag{5.2}$$

From equation (5.2) the dependence graph of figure 5.1.a can be derived. Each node in the graph performs a multiplication and an addition (figure 5.1.b). The computational complexity of the linear convolution in the time domain is the number of nodes in the dependence graph, which is $N_a N_x$.

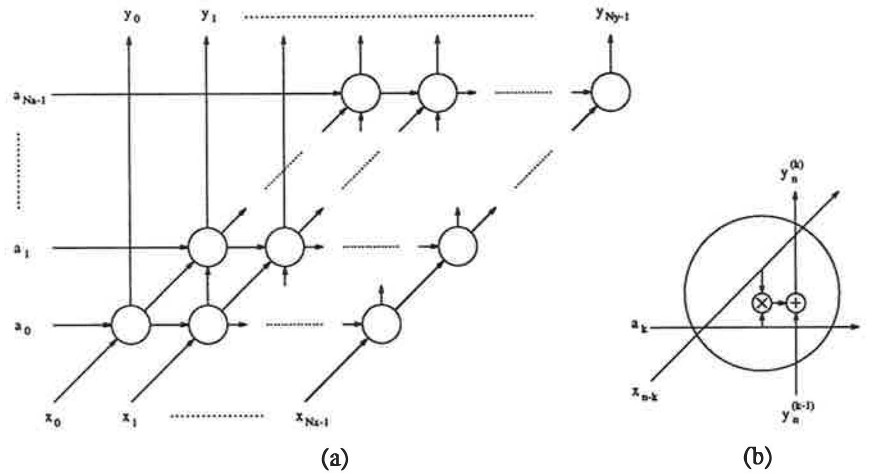


Figure 5.1: Dependence graph for convolution (a) and the elementary operation (b).

FFT based linear convolution

From a computational point of view it is more efficient to transform the sequences to frequency domain first using the DFT, where the computational complexity for convolution is much lower, and then perform an inverse transformation. Let **y** be given, then its N_y-point DFT is the scalar sequence $Y = [Y_0 \cdots Y_{N_y-1}]$, where Y_m is defined as

$$\mathcal{Y}_m = \sum_{n=0}^{N_y-1} y_n W_{N_y}^{nm} \tag{5.3}$$

and $W_{N_y} = \exp(j2\pi/N_y)$. Conversely, if Y is given then its N_y-point IDFT is **y**, where y_n is defined as

$$y_n = \frac{1}{N_y} \sum_{m=0}^{N_y-1} Y_m W_{N_y}^{-nm} \tag{5.4}$$

As in the previous chapter we can use the matrix notation of the DFT. Let $\mathbf{W}_{N_y}$ be the $N_y \times N_y$ DFT matrix, then the matrix notations of equations (5.3) and (5.4) are

$$\text{DFT:} \quad Y = \mathbf{y}\,\mathbf{W}_{N_y} \tag{5.5}$$

$$\text{IDFT:} \quad \mathbf{y} = \frac{1}{N_y} Y\,\mathbf{W}^*_{N_y} \tag{5.6}$$

Let A and X be defined as

$$A = \mathbf{a}[\,\mathbf{I}_{N_a}\,|\,0\,]\mathbf{W}_{N_y}, \qquad X = \mathbf{x}[\,\mathbf{I}_{N_x}\,|\,0\,]\mathbf{W}_{N_y} \tag{5.7}$$

then Y is the element-wise product

$$Y = A \otimes X = \left(\mathbf{a}[\,\mathbf{I}_{N_a}\,|\,0\,]\mathbf{W}_{N_y}\right) \otimes \left(\mathbf{x}[\,\mathbf{I}_{N_x}\,|\,0\,]\mathbf{W}_{N_y}\right) \tag{5.8}$$

The linear convolution result is now the N_y-point IDFT of Y

$$\mathbf{y} = \frac{1}{N_y} Y\,\mathbf{W}^*_{N_y} = \frac{1}{N_y}\left(\left(\mathbf{a}[\,\mathbf{I}_{N_a}\,|\,0\,]\mathbf{W}_{N_y}\right) \otimes \left(\mathbf{x}[\,\mathbf{I}_{N_x}\,|\,0\,]\mathbf{W}_{N_y}\right)\right)\mathbf{W}^*_{N_y} \tag{5.9}$$

If the DFTs of $\mathbf{a}$ and $\mathbf{x}$ are given then their linear convolution result in DFT domain can be computed using N_y multiplications, which is far less complex then computing the linear convolution in the time domain. However, the computation of the N_y-point DFTs and IDFT using the straight-forward matrix-vector multiplications in equation (5.9) requires N_y^2 multiplications. But if the DFT size would be a power of 2 we can use Fast Fourier Transform (FFT) to compute the DFT. In order to distinguish between the computation of the DFT directly and computation of the DFT by the FFT, we introduce the N-point FFT matrix $\mathbf{F}_N$. The N-point FFT matrix $\mathbf{F}_N$ is only defined if N is a power of 2. Let N be the smallest power of 2 larger or equal to N_y, then equation (5.9) can be represented as

$$\mathbf{y}[\,\mathbf{I}_{N_y}\,|\,0\,] = \frac{1}{N}\left(\left(\mathbf{a}[\,\mathbf{I}_{N_a}\,|\,0\,]\mathbf{F}_N\right) \otimes \left(\mathbf{a}[\,\mathbf{I}_{N_a}\,|\,0\,]\mathbf{F}_N\right)\right)\mathbf{F}^*_N \tag{5.10}$$

The computational complexity of an N-point (I)FFT is[1] $\frac{1}{2}N\log N$, so that the total computational complexity required to compute $\mathbf{y}$ using the FFT is then $\mathcal{O}(N\log N)$, whereas the computational complexity of the direct computation of $\mathbf{y} = \mathbf{a} * \mathbf{x}$ is $\mathcal{O}(N^2)$.

Thus, performing the convolution by using the FFT is far more efficient in terms of computational complexity, especially if N_a and N_x are large. However, a problem with large FFT sizes occurs when the convolution algorithm must be implemented in hardware. Commercially available FFT processors cannot cope with sequences having lengths larger than 1024 samples. A solution would be the use of DSP processor boards, but these have the disadvantage that they are not optimized in terms of memory management, hardware volume and power consumption. Therefore, in the next sections we will analyze the convolution algorithm and try to find an algorithm that can be mapped optimally onto available hardware architectures.

5.3 Sectioning of Large Convolution Problems

In the previous section we have considered the general description of convolution in the time domain and in the frequency domain. However, when vectors are too large, which can be the

[1]For conveniency, the base 2 logarithm $^2\log$ is denoted as log.

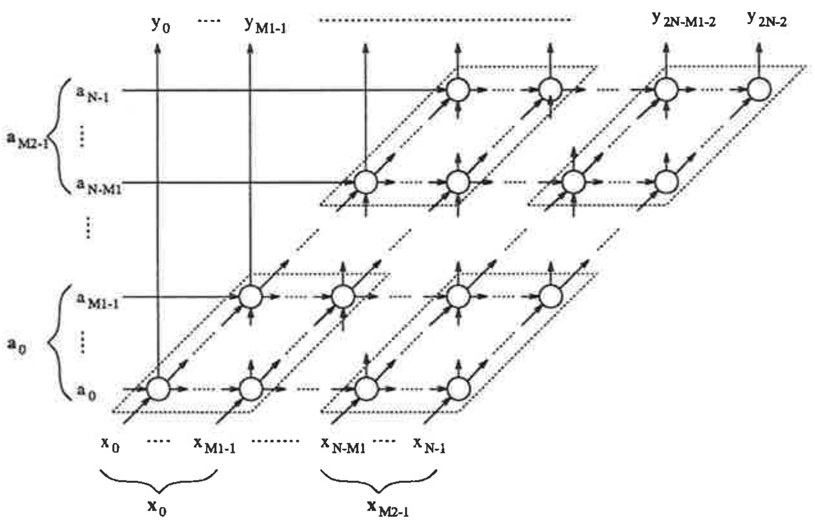


Figure 5.2: Dependence graph for convolution with clustering of $M_1 \times M_1$ nodes.

case in many applications, the direct implementations of the time domain or frequency domain algorithms will be inefficient and impractical, especially when dedicated architectures are used. Therefore we can use *overlap-add* or *overlap-discard* methods to section one or both sequences in smaller subsequences [OS89]. In this section we shall first focus on the overlap-add method. The overlap-discard method can be derived from the overlap-add method as we will show.

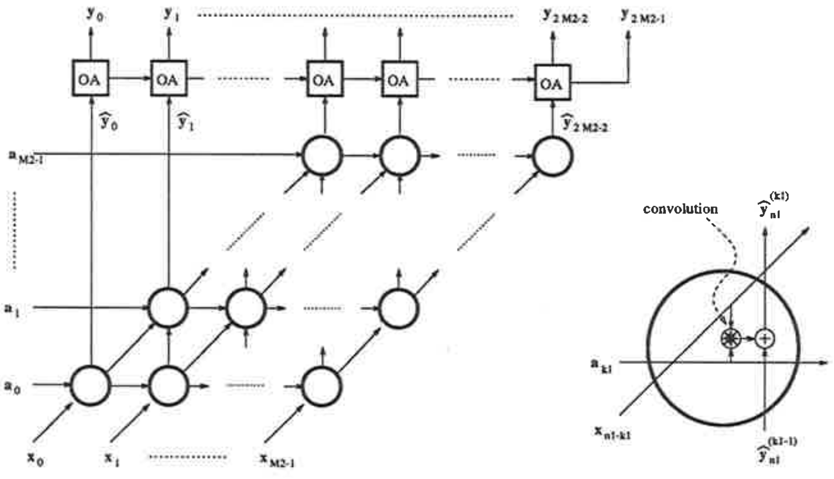
Block-signal processing: overlap-add

Let **a** and **x** be defined as in the previous section, but assume for convenience that $N_a = N_x = N$. The analysis can be adapted straightforwardly for the case that $N_a \neq N_x$. Furthermore assume that we have some convolution architecture for M_1-dimensional vectors only (either in time domain or frequency domain), with $M_1 \ll N$. For convenience we assume that N is a multiple of M_1, say $N = M_2 M_1$ and that M_1 is a power of 2. Then we can section our convolution problem as follows. Let for $m = 0, \cdots, M_2 - 1$ the M_1-dimensional vectors $\mathbf{a}_m$ and $\mathbf{x}_m$ be defined as

$$\mathbf{a}_m = [a_{mM_1}\ a_{mM_1+1}\ \cdots\ a_{(m+1)M_1-1}], \qquad \mathbf{x}_m = [x_{mM_1}\ x_{mM_1+1}\ \cdots\ a_{(m+1)M_1-1}]$$

From [AB74] we know that substitution of

$$\begin{aligned} k &\triangleq k_1 M_1 + k_2, \quad k_2 = 0, \cdots, M_1 - 1 \\ n &\triangleq n_1 M_1 + n_2, \quad n_2 = 0, \cdots, M_1 - 1 \end{aligned}$$



(a) (b)

Figure 5.3: Signal flow graph derived from the dependence graph (a) and the processing element (b). The blocks labeled with OA perform the overlap-add operation of the $2M_1 - 1$-dimensional vectors $\hat{\mathbf{y}}_n$. This operation is illustrated in figure D.1 in appendix D.

in equation (5.1) implies the application of the overlap-add convolution method. Hence

$$\begin{aligned} y_{n_1 M_1 + n_2} &= \sum_{k_1=0}^{M_2-1} \sum_{k_2=0}^{M_1-1} a_{k_1 M_1 + k_2} \; x_{n_1 M_1 + n_2 - (k_1 M_1 + k_2)} \\ &= \sum_{k_1=0}^{M_2-1} \sum_{k_2=0}^{M_1-1} a_{k_1 M_1 + k_2} \; x_{(n_1 - k_1) M_1 + n_2 - k_2} \end{aligned} \tag{5.11}$$

The interpretation in the dependence graph from figure 5.2 is as follows. Obviously equation (5.11) can be interpreted as the clustering of $M_1 \times M_1$ nodes. The arithmetic operation within a node is recognized as a convolution of two M_1-dimensional vectors of length M_1. Now we can derive a signal flow graph [Kun88] of the dependence graph, where we assume that the processing element performs the convolution operation within one cluster, see figure 5.3.a. The processing element is shown in figure 5.3.b. The blocks labeled with OA perform the overlap-add operation of the $2M_1 - 1$-dimensional vectors $\hat{\mathbf{y}}_n$. This operation is illustrated in figure D.1 in appendix D. The resulting vectors $\mathbf{y}_n$ are M_1-dimensional and are the non-overlapping sections of the vector

$$\mathbf{y} = \mathbf{a} * \mathbf{x} = [\mathbf{y}_0 \cdots \mathbf{y}_{2M_1-1}] \tag{5.12}$$

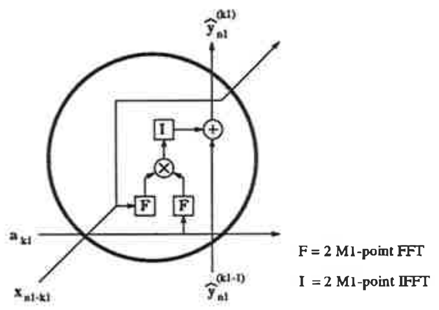


Figure 5.4: Processing element of the signal flow graph. The short convolution is performed in frequency domain.

Observe that the computational complexity of time domain convolution with overlap-add has not changed compared to the time domain convolution without overlap-add (if we assume that the computational complexity of the overlap-add operation is negligible). But if we use the FFT to perform the short convolution, then it becomes more efficient. Since M_1 is a power of 2, the nearest power of 2 to $2M_1 - 1$ is $2M_1$, so that we can draw an alternative processing element as in figure 5.4. The computational complexity of this node is given as $M_1(2+3\log 2M_1)$. Hence, the overall computational complexity is given as $M_1 M_2^2(2+3\log 2M_1)$, which is lesser than the computational complexity of the direct linear convolution $N^2 = M_1^2 M_2^2$ (where we assume that $M_1 > 2 + 3\log 2M_1$, implying that $M_1 \geq 32$). However, the computational complexity is still more than the long FFT domain convolution. The signal flow graph of figure 5.3.a represents the usual way to implement the overlap-add convolution method.

We can now optimize the signal flow graph by the observation that within the processor element of figure 5.4 all vectors $\mathbf{a}_m$ and $\mathbf{x}_m$ are transformed M_2 times, respectively. However, it is more efficient to make use of the linearity of the FFT operation, which implies that we can transform each vector once at the input side. In the same way we can inverse transform each vector $\hat{\mathbf{y}}_m$ at the output side. This results in the signal flow graph in figure 5.5.a, which is equivalent to the signal flow graph in figure 5.3.a, but requires far less transformations. The arithmetic operation within a processing element is now only an element-wise multiplication in stead of a convolution.

If we analyze the structure of the signal flow graph of figure 5.5.a we recognize that the kernel has exactly the same structure as the original dependence graph. Moreover, since the multiplication operator within the processing element operates element-wise, the complete kernel as a whole operates element-wise. Let the vectors $\boldsymbol{A}_m$, $\boldsymbol{X}_m$ and $\widehat{\boldsymbol{Y}}_m$ be the $2M_1$-point FFTs of the vectors $\mathbf{a}_m$, $\mathbf{x}_m$ and $\hat{\mathbf{y}}_m$, respectively (see figure 5.5.a). Then the kernel can be

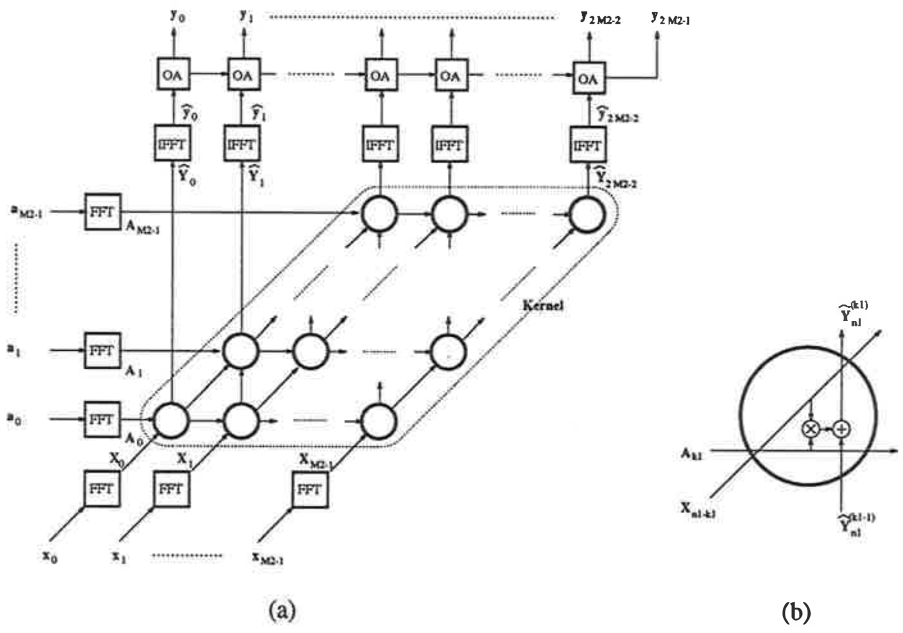


Figure 5.5: Signal flow graph of the overlap-add in frequency domain (a) and the processing element (b).

represented as

$$\widehat{\boldsymbol{Y}}_m = \sum_{n=0}^{M_2-1} \boldsymbol{A}_n \otimes \boldsymbol{X}_{m-n} \tag{5.13}$$

From the signal flow graph we see that the number of $2M_1$-points (I)FFTs is given as $4M_2 - 1$. The number of multiplications within the kernel is given as $2M_2^2 M_1$. Hence, the overall computational complexity is given by $M_1(4M_2 - 1)\log 2M_1 + 2M_1M_2^2$.

Relation to the linear convolution network

At this moment we can easily explain the relationship with the multirate convolution system. For $m = 0, \cdots, M_2 - 1$ define $[\boldsymbol{A}_m]_k \triangleq A_{m,k}$ and define the $2M_1 \times 2M_1$ diagonal matrix

$$\mathbf{D}_m \triangleq \operatorname{diag}\{A_{m,0}, \cdots, A_{m,2M_1-1}\} \tag{5.14}$$

Then equation (5.13) can be expressed using the diagonal matrices $\mathbf{D}_m$

$$\widehat{\boldsymbol{Y}}_m = \sum_{n=0}^{M_2-1} \boldsymbol{X}_m \mathbf{D}_{m-n} \tag{5.15}$$

The advantage of equation (5.15) is that we can write it in matrix form

$$\left[\widehat{\boldsymbol{Y}}_0 \cdots \widehat{\boldsymbol{Y}}_{2M_2-2}\right] = [\boldsymbol{X}_0 \cdots \boldsymbol{X}_{M_2-1}] \begin{bmatrix} \mathbf{D}_0 & \cdots & \mathbf{D}_{M_2-1} & & 0 \\ & \ddots & & \ddots & \\ 0 & & \mathbf{D}_0 & \cdots & \mathbf{D}_{M_2-1} \end{bmatrix} \tag{5.16}$$

Observe that equation (5.16) represents an ordinary matrix-vector multiplication. However, in this form it can also be interpreted as a straight-forward representation of a block signal processing application (namely overlap-add convolution) in matrix form.

Now define the $K \times KL$ matrix $\zeta_{K,L}$ as

$$\zeta_{K,L} = \left[\mathbf{I}_K \mid z^{-1}\mathbf{I}_K \mid \cdots \mid z^{-(L-1)}\mathbf{I}_K\right]$$

Then from equation (5.16) a z-domain representation can be derived as

$$\begin{aligned} \widehat{\boldsymbol{Y}}(z) &\triangleq \left[\widehat{\boldsymbol{Y}}_0 \cdots \widehat{\boldsymbol{Y}}_{2M_2-2}\right] \zeta^t_{2M_1,2M_2-1} \\ &= [\boldsymbol{X}_0 \cdots \boldsymbol{X}_{M_2-1}] \begin{bmatrix} \mathbf{D}_0 & \cdots & \mathbf{D}_{M_2-1} & & 0 \\ & \ddots & & \ddots & \\ 0 & & \mathbf{D}_0 & \cdots & \mathbf{D}_{M_2-1} \end{bmatrix} \zeta^t_{2M_1,2M_2-1} \\ &= \left([\boldsymbol{A}_0 \cdots \boldsymbol{A}_{M_2-1}]\, \zeta^t_{2M_1,M_2}\right) \otimes \left([\boldsymbol{X}_0 \cdots \boldsymbol{X}_{M_2-1}]\, \zeta^t_{2M_1,M_2}\right) \\ &= \left(\sum_{m=0}^{M_2-1} \boldsymbol{A}_m z^{-m}\right) \otimes \left(\sum_{m=0}^{M_2-1} \boldsymbol{X}_m z^{-m}\right) \\ &\triangleq \boldsymbol{A}_0(z) \otimes \boldsymbol{X}_0(z) \end{aligned} \tag{5.17}$$

The inverse FFT of $\widehat{\boldsymbol{Y}}(z)$ is

$$\frac{1}{2M_1}\widehat{\boldsymbol{Y}}(z)\mathbf{F}^*_{2M_1} = \frac{1}{2M_1}(\boldsymbol{A}_0(z) \otimes \boldsymbol{X}_0(z))\mathbf{F}^*_{2M_1} \triangleq \hat{\mathbf{y}}(z) \tag{5.18}$$

But this equation is equivalent to equation (4.24) in section 4.4 if we consider the fact that $\mathbf{F}_{2M_1} \equiv \mathbf{W}_{2M_1}$.

Following the steps in section 4.4 this equation results in the multirate convolution system based on the linear convolution network with the DFT, given in figure 4.6.a. In fact, figure 5.5 and figure 4.6.a are equivalent. Moreover, reconsider that $\boldsymbol{A}_0(z)$ and $\boldsymbol{X}_0(z)$ are

$$\boldsymbol{A}_0(z) = \mathbf{a}(z)[\,\mathbf{I}_{M_1} \mid 0\,]\mathbf{F}_{2M_1}, \qquad \boldsymbol{X}_0(z) = \mathbf{x}(z)[\,\mathbf{I}_{M_1} \mid 0\,]\mathbf{F}_{2M_1} \tag{5.19}$$

where

$$\mathbf{a}(z) = \sum_{m=0}^{M_2-1} \mathbf{a}_m z^{-m} \equiv [a_0(z) \cdots a_{M_2-1}(z)] \tag{5.20}$$

$$\mathbf{x}(z) = \sum_{m=0}^{M_2-1} \mathbf{x}_m z^{-m} \equiv [x_0(z) \cdots x_{M_2-1}(z)] \tag{5.21}$$

In equations (5.20) and (5.21) again the polyphase components $a_m(z)$ and $x_m(z)$ appear! The second last line of equation (5.17) can then be interpreted as a polyphase decomposition of **a** and **x** represented as an order $M_2 - 1$ polynomial in z with $2M_1$-dimensional vector coefficients. So we have found a formal way to describe the multirate convolution network of the previous chapter in terms of blocks ($2M_1$-dimensional vectors), but also the way back. This novel result is extremely important in understanding multirate convolution systems in terms of processing behavior (by means of the signal flow graph representation).

Block-signal processing: overlap-discard

We have now shown that the figure 5.5 and figure 4.6.a are both representations of the same realizations of the overlap-add convolution algorithm. But then it must be possible to give a signal flow graph from the realizations of the overlap-discard convolution algorithm as given in figure 4.6.b. As a starting point we take equation (4.27), in which we have introduced the FFT matrix

$$\hat{\mathbf{y}}(z) = \frac{1}{2M_2}\left(\mathbf{A}_0(z) \otimes \widehat{\mathbf{X}}(z)\right)\mathbf{F}^*_{2M_2} \tag{5.22}$$

Recall equation (4.29), then we have

$$\widehat{\mathbf{X}}(z) = \hat{\mathbf{x}}(z)\mathbf{F}_{2M_2} = \mathbf{x}(z)[\, z^{-1}\mathbf{Z}_{M_1} \mid \mathbf{I}_{M_1}\,]\mathbf{F}_{2M_2} \tag{5.23}$$

In order to derive a signal flow graph substitute equation (5.21) into the expression for $\hat{\mathbf{x}}(z)$, which gives

$$\begin{aligned}\hat{\mathbf{x}}(z) &= \left(\sum_{m=0}^{M_2-1} \mathbf{x}_m z^{-m}\right)[\, z^{-1}\mathbf{Z}_{M_1} \mid \mathbf{I}_{M_1}\,] \\ &= \sum_{m=0}^{M_2-1}[\, z^{-1}\mathbf{Z}_{M_1}\mathbf{x}_m \mid \mathbf{x}_m\,]z^{-m} \\ &= \left[\sum_{m=1}^{M_2} \mathbf{Z}_{M_1}\mathbf{x}_{m-1}z^{-m} \;\middle|\; \sum_{m=0}^{M_2-1}\mathbf{x}_m z^{-m}\right]\end{aligned} \tag{5.24}$$

Define for $m = 0, \cdots, M_2$ the $2M_1$ vector $\widehat{\mathbf{x}}_m$ as

$$\widehat{\mathbf{x}}_m \triangleq \begin{cases} [\,0 \mid \mathbf{x}_0\,] & \text{if } m = 0 \\ [\,\mathbf{Z}_{M_1}\mathbf{x}_{M_2-1} \mid 0\,] & \text{if } m = M_2 \\ [\,\mathbf{Z}_{M_1}\mathbf{x}_{m-1} \mid \mathbf{x}_m\,] & \text{otherwise} \end{cases} \tag{5.25}$$

and substitute $\widehat{\mathbf{x}}_m$ into equation (5.24) then we get

$$\widehat{\mathbf{x}}(z) = \sum_{m=0}^{M_2} \widehat{\mathbf{x}}_m z^{-m} \tag{5.26}$$

Let us follow the reversed derivation of equation (5.17) for $A_0(z) \otimes \widehat{X}(z)$, which should lead to a similar expression as in equation (5.16):

$$\begin{aligned} A_0(z) \otimes \widehat{X}(z) &= \left(\sum_{m=0}^{M_2-1} A_m z^{-m}\right) \otimes \left(\sum_{m=0}^{M_2} \widehat{X}_m z^{-m}\right) \\ &= \left([A_0 \cdots A_{M_2-1}]\zeta^t_{2M_1,M_2}\right) \otimes \left([\widehat{X}_0 \cdots \widehat{X}_{M_2}]\zeta^t_{2M_1,M_2+1}\right) \\ &= \left[\widehat{X}_0 \cdots \widehat{X}_{M_2}\right] \begin{bmatrix} \mathbf{D}_0 & \cdots & \mathbf{D}_{M_2-1} & & 0 \\ & \ddots & & \ddots & \\ 0 & & \mathbf{D}_0 & \cdots & \mathbf{D}_{M_2-1} \end{bmatrix} \zeta^t_{2M_1,2M_2} \\ &\triangleq \left[\widehat{\widehat{Y}}_0 \cdots \widehat{\widehat{Y}}_{2M_2-1}\right] \zeta^t_{2M_1,2M_2} \end{aligned} \tag{5.27}$$

Combining this result with equation (5.22), then for $m = 0, \cdots, 2M_2 - 1$ we can define the $2M_1$-dimensional vector

$$\begin{aligned} \widehat{\mathbf{y}}(z) &= \frac{1}{2M_1}\left[\widehat{\widehat{Y}}_0 \cdots \widehat{\widehat{Y}}_{2M_2-1}\right] \zeta^t_{2M_1,2M_2}\mathbf{F}^*_{2M_1} \\ &= \left[\frac{1}{2M_1}\widehat{\widehat{Y}}_0\mathbf{F}^*_{2M_1} \cdots \frac{1}{2M_1}\widehat{\widehat{Y}}_{2M_2-1}\mathbf{F}^*_{2M_1}\right] \zeta^t_{2M_1,2M_2} \\ &= \left[\widehat{\mathbf{y}}_0 \cdots \widehat{\mathbf{y}}_{2M_2-1}\right] \zeta^t_{2M_1,2M_2} \end{aligned} \tag{5.28}$$

Moreover, applying equation (5.22) results in the expression for $\mathbf{y}(z)$

$$\begin{aligned} \mathbf{y}(z) &= \left[\widehat{\mathbf{y}}_0 \cdots \widehat{\mathbf{y}}_{2M_2-1}\right] \zeta^t_{2M_1,2M_2} \left[\frac{0}{\mathbf{I}_{M_1}}\right] \\ &= \left[\widehat{\mathbf{y}}_0 \left[\frac{0}{\mathbf{I}_{M_1}}\right] \cdots \widehat{\mathbf{y}}_{2M_2-1}\left[\frac{0}{\mathbf{I}_{M_1}}\right]\right] \zeta^t_{M_1,2M_2} \\ &\equiv \left[\mathbf{y}_0 \cdots \mathbf{y}_{2M_2-1}\right] \zeta^t_{M_1,2M_2} = \mathbf{y}\,\zeta^t_{M_1,2M_2} \end{aligned} \tag{5.29}$$

Relation to the circular convolution network

We can summarizing the results for the overlap-discard convolution algorithm as follows. The kernel of the signal flow graph is determined by equation (5.27), where we recognize the matrix notation of a block convolution structure as in equation (5.16), given as

$$\left[\widehat{\widehat{\boldsymbol{Y}}}_0 \cdots \widehat{\widehat{\boldsymbol{Y}}}_{2M_2-1}\right] = \left[\widehat{\boldsymbol{X}}_0 \cdots \widehat{\boldsymbol{X}}_{M_2}\right] \begin{bmatrix} \mathbf{D}_0 & \cdots & \mathbf{D}_{M_2-1} & & 0 \\ & \ddots & & \ddots & \\ 0 & & \mathbf{D}_0 & \cdots & \mathbf{D}_{M_2-1} \end{bmatrix} \tag{5.30}$$

In fact, the kernel structure is the same as in figure 5.5.a. The only difference is that the dimension in the horizontal direction is increased by one. For $m = 0, \cdots, M_2$, the vectors $\widehat{\boldsymbol{X}}_m$ are the $2M_1$-point FFTs of $\hat{\mathbf{x}}_m$ which is related to $\mathbf{x}_m$ according to equation (5.25). In practice this means that $\hat{\mathbf{x}}_m$ is

$$\hat{\mathbf{x}}_m \triangleq \begin{cases} [0 \cdots 0\ x_0 \cdots x_{M_1-1}] & \text{if } m = 0 \\ [x_{N-M_1} \cdots x_{N-1}\ 0 \cdots 0] & \text{if } m = M_2 \\ \left[x_{(m-1)M_1} \cdots x_{mM_1-1}\ x_{mM_1} \cdots x_{(m+1)M_1-1}\right] & \text{otherwise} \end{cases} \tag{5.31}$$

From equations (5.28) and (5.29) we can determine $\mathbf{y}_m$, for $m = 0, \cdots, 2M_2 - 1$, from $\widehat{\widehat{\boldsymbol{Y}}}_m$ as follows

$$\mathbf{y}_m = \frac{1}{2M_2} \widehat{\widehat{\boldsymbol{Y}}}_m \mathbf{F}^*_{2M_1} \left[\frac{0}{\mathbf{I}_{M_1}} \right] \tag{5.32}$$

In words, perform the $2M_1$-point IFFT of $\boldsymbol{Y}_m$ and discard the first M_1 entries of the IFFT result. The signal flow graph of the overlap-discard convolution algorithm can now be determined as in figure 5.6, with processor element as in figure 5.5.b. From the signal flow graph we see that the number of $2M_1$-points (I)FFTs is given as $4M_2 + 1$ which is 2 more than required for overlap-add. The number of multiplications within the kernel is given as $2M_1M_2(M_2 + 1)$, which is $2M_1M_2$ more than required for overlap-add. Hence, the overall computational complexity for overlap-discard is given as $M_1(4M_2 + 1)\log 2M_1 + 2M_1M_2(M_2 + 1)$. On first sight the overlap-discard convolution algorithm is somewhat more expensive than the overlap-add algorithm in terms of computational complexity. However, in the next sections we will show that the additional overhead due to the actual overlap-add operation (the "OA"-block in figure 5.5.a) is more expensive than the additional overhead due to the actual overlap-discard operation (the "OD"-block in figure 5.6) in terms of hardware.

5.4 Short Length FFT Processing

In this section we consider the kernel of both signal flow graphs of the overlap-add and the overlap-discard. Observe that the kernels of both signal flow graphs have the same structure.

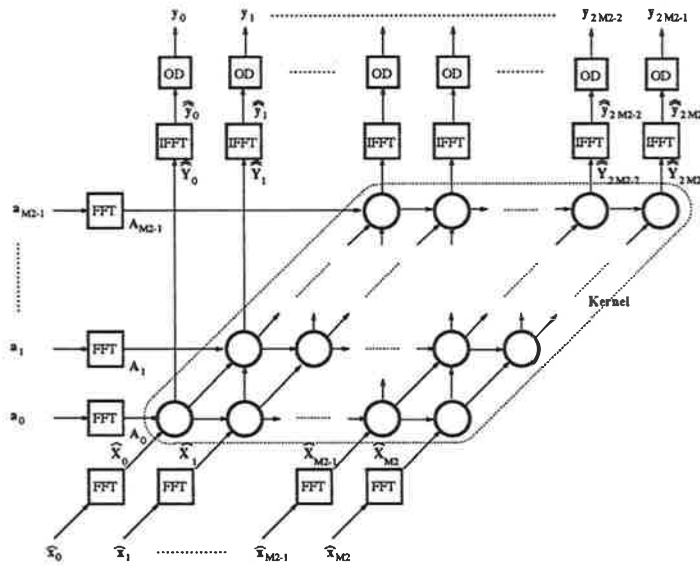


Figure 5.6: Signal flow graph of the overlap-discard in frequency domain. The blocks labeled with OD perform the discard operation of the $2M_1 - 1$-dimensional vectors $\hat{\mathbf{y}}_n$. This operation is illustrated in figure D.2 in appendix D.

The only difference is the dimension: the signal flow graph of the overlap-add contains M_2^2 processor elements, the signal flow graph of the overlap-discard contains $M_2(M_2+1)$ processor elements. However, in both we recognize a linear convolution-like structure, which is confirmed if we reconsider equations (5.16) and (5.30). For this reason it is possible to implement the FFT matrix to perform the kernel operation efficiently. We will show this for the overlap-add kernel only, since the structure is the same for the overlap-discard kernel.

Recall equation (5.13) and observe that it can be written as

$$\begin{aligned}\left[\hat{Y}_{m,0} \cdots \hat{Y}_{m,2M_1-1}\right] &= \sum_{n=0}^{M_2-1} [A_{n,0} \cdots A_{n,2M_1-1}] \otimes [X_{m-n,0} \cdots X_{m-n,2M_1-1}] \\ &= \left[\sum_{n=0}^{M_2-1} A_{n,0}X_{m-n,0} \cdots \sum_{n=0}^{M_2-1} A_{n,2M_1-1}X_{m-n,2M_1-1}\right] \qquad (5.33)\end{aligned}$$

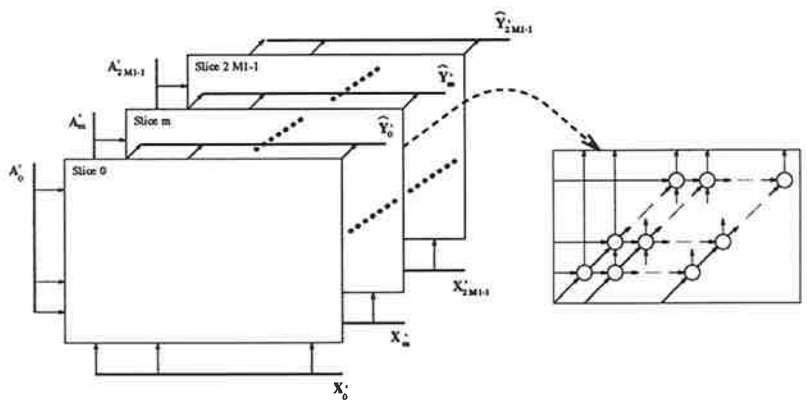


Figure 5.7: Kernel of the signal flow graph of the overlap-add in frequency domain and the processing element. The slices represent signal flow graph of a convolution.

But we can also construct an $2M_2 - 1 \times 2M_1$ matrix

$$\left[\widehat{\boldsymbol{Y}}^t_0 \cdots \widehat{\boldsymbol{Y}}^t_{2M_2-2}\right] = \begin{bmatrix} \widehat{Y}_{0,0} & \cdots & \widehat{Y}_{2M_2-2,0} \\ \vdots & & \vdots \\ \widehat{Y}_{0,2M_1-1} & \cdots & \widehat{Y}_{2M_2-2,2M_1-1} \end{bmatrix} \triangleq \begin{bmatrix} \widehat{\boldsymbol{Y}}'_0 \\ \vdots \\ \widehat{\boldsymbol{Y}}'_{2M_1-1} \end{bmatrix} \tag{5.34}$$

Hence, the vectors $\widehat{\boldsymbol{Y}}'_m$ are convolution results

$$\left[\widehat{\boldsymbol{Y}}'_m\right]_k = \widehat{Y}_{k,m} = \sum_{n=0}^{M_2-1} A_{n,m} X_{k-n,m} \tag{5.35}$$

Now for $m = 0, \cdots, M_2 - 1$ define the vectors

$$\boldsymbol{A}'_m = [A_{m,0} \cdots A_{M_2-1,0}], \qquad \boldsymbol{X}'_m = [X_{m,0} \cdots X_{M_2-1,0}]$$

then

$$\widehat{\boldsymbol{Y}}'_m = \boldsymbol{A}'_m * \boldsymbol{X}'_m \tag{5.36}$$

Now we can draw the kernel of the signal flow graph in figure 5.5 as in figure 5.7, where each m^{th} *slice* represents the m^{th} linear convolution.

This result can also be derived more formally from the matrix notation of equation (5.16). Observe that we can define a $2M_1(2M_2 - 1) \times 2M_1(2M_2 - 1)$ permutation matrix[2] $\mathbf{P}_y$ and a

[2]Permutation matrices have the property that they are orthogonal.

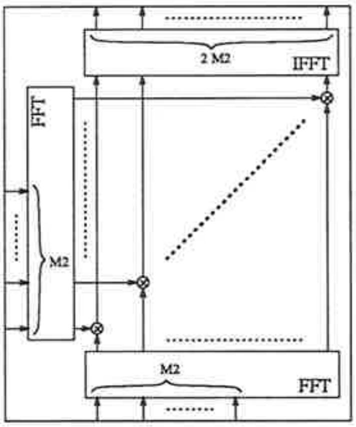


Figure 5.8: The frequency domain description of one slice from the kernel of the signal flow graph of the linear convolution in frequency domain with overlap-add.

$M_1 M_2 \times M_1 M_2$ permutation matrix $\mathbf{P}_x$, such that

$$\left[\widehat{\boldsymbol{Y}}'_0 \cdots \widehat{\boldsymbol{Y}}'_{2M_1-1}\right] = \left[\widehat{\boldsymbol{Y}}_0 \cdots \widehat{\boldsymbol{Y}}_{2M_2-2}\right] \mathbf{P}_y \tag{5.37}$$

$$\left[\boldsymbol{X}'_0 \cdots \boldsymbol{X}'_{M_1-1}\right] = \left[\boldsymbol{X}_0 \cdots \boldsymbol{X}_{M_2-1}\right] \mathbf{P}_x \tag{5.38}$$

Combining equations (5.16) and (5.38) gives[3]

$$\begin{aligned}
\left[\widehat{\boldsymbol{Y}}'_0 \cdots \widehat{\boldsymbol{Y}}'_{2M_1-1}\right] &= \left[\boldsymbol{X}_0 \cdots \boldsymbol{X}_{M_2-1}\right] \mathbf{P}_x \begin{bmatrix} \mathbf{D}_0 & \cdots & \mathbf{D}_{M_2-1} & & 0 \\ & \ddots & & \ddots & \\ 0 & & \mathbf{D}_0 & \cdots & \mathbf{D}_{M_2-1} \end{bmatrix} \mathbf{P}_y^t \\
&= \left[\boldsymbol{X}'_0 \cdots \boldsymbol{X}'_{M_2-1}\right] \begin{bmatrix} \begin{bmatrix} A_{0,0} & \cdots & A_{0,2M_1-1} & & 0 \\ & \ddots & & \ddots & \\ 0 & & A_{0,0} & \cdots & A_{0,2M_1-1} \end{bmatrix} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \begin{bmatrix} A_{0,0} & \cdots & A_{0,2M_1-1} & & 0 \\ & \ddots & & \ddots & \\ 0 & & A_{0,0} & \cdots & A_{0,2M_1-1} \end{bmatrix} \end{bmatrix} \\
&= \left[\boldsymbol{A}'_0 * \boldsymbol{X}'_0 \cdots \boldsymbol{A}'_{M_2-1} * \boldsymbol{X}'_{M_2-1}\right]
\end{aligned} \tag{5.39}$$

[3]The large matrix in the second line of equation (5.39) is block-diagonal.

This linear convolution operation can again be performed efficiently using the FFT matrix. Assume again for convenience that M_2 is a power of 2, then the nearest power of 2 to $2M_2 - 1$ is $2M_2$. Then equation (5.36) can equivalently be performed as follows

$$\widehat{\boldsymbol{Y}}'_m = \frac{1}{2M_2} \left(\left(\boldsymbol{A}'_m [\, \mathbf{I}_{M_2} \mid 0 \,] \mathbf{F}_{2M_2} \right) \otimes \left(\boldsymbol{X}'_m [\, \mathbf{I}_{M_2} \mid 0 \,] \mathbf{F}_{2M_2} \right) \right) \mathbf{F}^*_{2M_2} \tag{5.40}$$

This results in a kernel as it is shown in figure 5.8. Embedding the kernel within the overlap-add signal flow graph results in the three-dimensional signal flow graph of figure 5.9, in which each polygon represents a $2M_1$-point or $2M_2$-point (I)FFT.

Some notes on computational complexity

The computational complexity of this algorithm can now be determined as follows. The number of $2M_1$-point (I)FFTs is still given as $4M_2 - 1$. Observe that the kernel operation contains $2M_1$ slices, where each slice consists of three $2M_2$-point (I)FFTs and $2M_2$ multiplications. Hence, the overall computational complexity is then given as $M_1(4M_2 - 1)\log 2M_1 + 6M_1M_2\log 2M_2 + 4M_1M2$ for the overlap-add algorithm and $M_1(4M_2+1)\log 2M_1 + 6M_1M_2\log 2M_2 + 4M_1M2$ for the overlap-discard algorithm.

We can now summarize the expressions for the subsequent algorithms that we have developed. We assume that the worst-case realization of the linear convolution algorithm is the time domain convolution, which requires $M_1^2M_2^2$ operations, and the best-case realization is the frequency domain convolution which requires $3M_1M_2\log(2M_1M_2) + 2M_1M_2$ computations.[4] In table 5.1 the computational complexities of the subsequent algorithms are listed. Algorithm A refers to the signal flow graph in figure 5.3.a with processing element in figure 5.4. Algorithm B refers to the signal flow graph in figure 5.5.a. Algorithm C refers to the three-dimensional signal flow graph in figure 5.9.

In figure 5.10 the computational complexities are plotted as function of $N = M_1M_2$ (for convenience only the overlap-add case is shown). It appears that algorithm C requires twice as much operations as the best-case, independently of the convolution length (figure 5.10.a). However, for sequences which have lengths $N < 2^{16}$ algorithm B has less computational complexity. The reason is that if M_1 or M_2 is large, then algorithms A and B will perform as the best-case algorithm. In the limit case that $M_1 = N$ or $M_2 = 1$, we have that these algorithms are equivalent, i.e., they are all representations of the one-dimensional frequency domain convolution. A different case is if we require that $M_1 = M_2$ (figure 5.10.b). In that case we see that algorithm C requires far less operations than algorithms B and C for all N.

[4]This statement is not valid for $N = M_1M_2 < 8$. Therefore we will assume that $N = M_1M_2 \geq 8$.

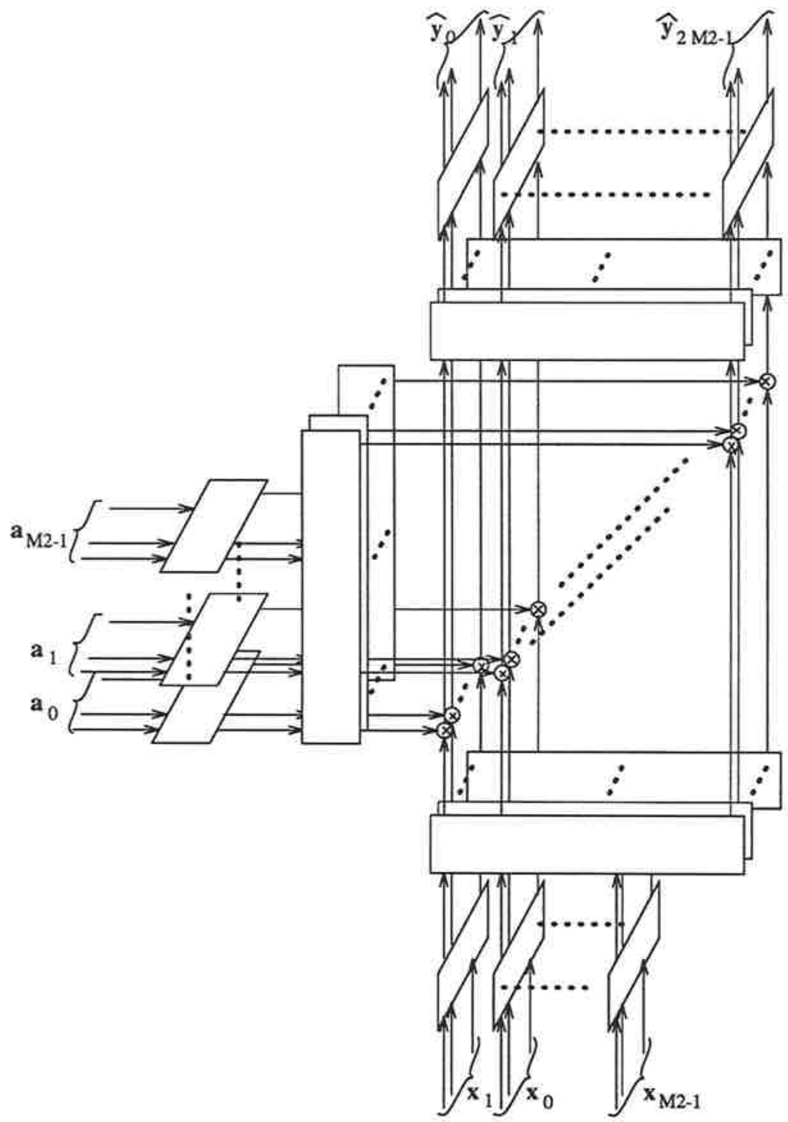
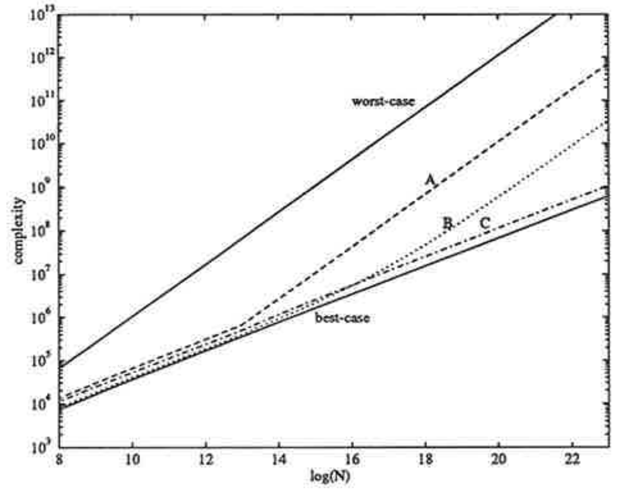
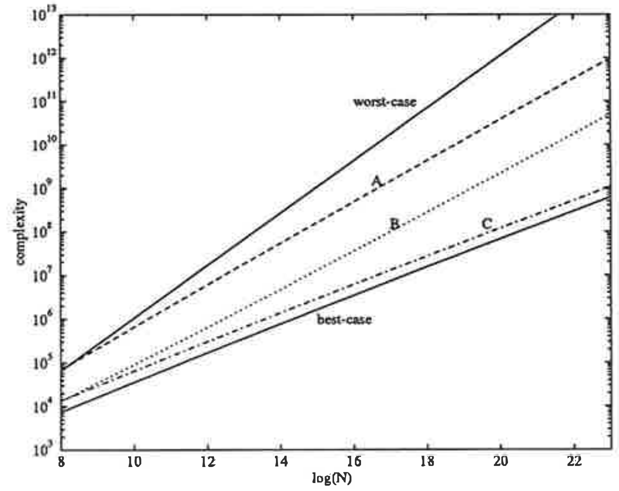


Figure 5.9: Three-dimensional signal flow graph of a convolution algorithm that consists of FFTs and multiplications. Instead of long FFTs in one-dimension short FFTs are used in two-dimensions. The polygons represent the FFTs. For convenience the overlap-add operation is omitted.



(a)



(b)

Figure 5.10: Computational complexity as function of $\log N$, where we used the expressions of table 5.1. In the plots only the overlap-add case is shown. In (a) we did not have conditions for M_1 and M_2, i.e. the computational complexities for each possible N is minimized over all combinations of $\{M_1, M_2\}$. In (b) we have introduced condition that $M_1 = M_2$.

	Computational Complexity
worst-case	$M_1^2 M_2^2$
overlap-add A	$M_1 M_2^2(2+3\log 2M_1)$
overlap-discard A	$M_1 M_2(M_2+1)(2+3\log 2M_1)$
overlap-add B	$M_1(4M_2-1)\log 2M_1 + 2M_1 M_2^2$
overlap-discard B	$M_1(4M_2+1)\log 2M_1 + 2M_1 M_2^2$
overlap-add C	$M_1(4M_2-1)\log 2M_1 + 2M_1 M_2 \log 2M_2 + 4M_1 M_2$
overlap-discard C	$M_1(4M_2+1)\log 2M_1 + 2M_1 M_2 \log 2M_2 + 4M_1 M_2$
best-case	$3M_1 M_2 \log 2M_1 M_2 + 2M_1 M_2$

Table 5.1: Computational complexities of the subsequent realizations of the linear convolution. The convolution length is given as $N = M_1 M_2$, where M_1 and M_2 are powers of 2.

5.5 Concluding Remarks

In this chapter we have used the graph manipulations to elaborate a formal relation between multirate convolution systems and block-signal processing. The starting point was the dependence graph of the long convolution problem. The dependence graph is projected onto two slightly different signal flow graphs, which are in fact graphical representation of the overlap-add and overlap-discard methods. We have shown, by means of formal "algebraic manipulations" and by means of illustrative "graph manipulations", that this representation is equivalent to the optimized multirate convolution system, which has been obtained in the previous chapter. However, we prefer the representation from a multirate point of view rather than the block-signal processing point of view. The multirate point of view allows us to combine our processing structures with novel multirate processing schemes, such as has been done in e.g. [CPR89, GV92]. In chapter 7 we propose some novel directions for future research in this area. It remains now to show the power of the graphical representation in designing hardware architectures. This will be the subject of the next chapter.

CHAPTER 6

HARDWARE ARCHITECTURE DESIGN

6.1 Introduction

We ended the previous chapters with a schematic description of the multirate convolution system. In this chapter we show that the multirate convolution system, indeed, can be mapped easily into efficient parallel hardware architectures. The specifications of an "efficient" hardware architecture are determined by the application. In our case it will be part of a real-time on-board SAR processor, and perform either the range or azimuth compression, see chapter 3. From the application point of view, we derive the following requirements of the architecture:

- The processor size and power consumption must be small. Especially for on-board satellite processing, these factors are hard limits.
- The input signals have lengths up to 5 Ksamples. In the future, however, signal lengths will even grow. We set the maximum convolution on 8 Ksamples.
- Effective data rates vary from 2.3 Msamples/sec up to 10 Msamples/sec. Our design goal is thus 10 Msamples/sec, but the system must be suitable for higher data rates in the future.

Nowadays it is customary to use off-the-shelf DSP components to solve long convolution problems. They have the advantage of being programmable and thus flexible. They are often specified by benchmarks, such as the processing time of an N-point FFT or the number of MOPS (Mega Operations per Seconds) or GOPS (Giga Operations per Seconds). A major disadvantage, however, is that they require a complex infrastructure, which includes I/O control and protocols, high speed data busses, high speed memory and memory control, intermediate buffering, etc.. Due to the required infrastructure, board size and power consumption are substantial. Moreover, increase of the number of processors does not lead to a linear increase in processing capacity. The extra overhead required for the multi-DSP infrastructure is due to the decline of the ideal linear increment. Our experience is that DSP boards are best specified by a trade-off between their benchmarks and overhead due to the required infrastructure. Doing so, we conclude that the smallest DSP board solution that meets the specifications listed above, contains an equivalent size of six Euro 6 board, and has $\mathcal{O}(100)$ W power consumption [Cat92].

Therefore we propose an alternative, namely a VLSI implementation of the multirate convolution system. Due to the structured data flow of the multirate convolution system, we can focus on the accuracy and the dynamic range, which are important aspects in SAR processing. The result will be a floating point single-chip convolution processor. It can perform an 8K convolution with an effective data rate of 2 Msamples/sec, thus the required 10 Msamples/sec effective data rate can be achieved with only 5 chips. The additional overhead due to the parallelization of the processor is negligible. A remarkable side-effect is that our VLSI implementation requires twice as much computations compared to a straight-forward implementation in DSP boards. Nevertheless, it is cheaper in terms of overall performance, volume and power consumption and, in many cases, also in terms of accuracy and dynamic range.

Essential in the design trajectory of the intended VLSI architecture, is the design of a prototype architecture. A prototype architecture is based on off-the-shelf components. It is used to solve typical hardware architecture design problems, such as balancing between I/O data rate and processing data rates, determination of a trade-off between processing flexibility and performance, modularity and parallelization possibilities.

In section 6.2 we give an algorithmic specification of the convolution schemes developed in the previous section. In section 6.3 we express the hardware specifications in terms of processing parameters, such as data rate, signal length, required memory. In section 6.4 we specify a dedicated VLSI architecture for the convolution system. Here we show that the multirate convolution system can be mapped onto a single-chip hardware architecture.

6.2 Algorithm Specification

The algorithms that we consider are represented in algorithms 1 and 2, see pages 84 and 85. We assume the more general case than in the previous chapter. Let $\mathbf{a}$ and $\mathbf{x}$ be the sequences to be convolved, with lengths N_a and N_x, respectively, and let $\mathbf{y} = \mathbf{a} * \mathbf{x}$ be their convolution. Given M_1 as in section 5.3. We assume that $N_a = M_a M_1$ and $N_x = M_x M_1$ with M_1 a power of 2. In case of SAR processing, the reference can be either the pulse replica or the azimuth filter, and the data is either the pulse echo signal or the azimuth echo signal. We will use a $2M_1$-point (I)FFT and a $2M_2$-point (I)FFT, where M_2 is also a power of 2. Moreover, we assume that M_a and M_x are chosen such that $2M_2$ is the nearest power of 2 to both $M_a + M_x - 1$ and $M_a + M_x$, i.e., we can use identical FFT sizes in both overlap-add and overlap-discard algorithms.

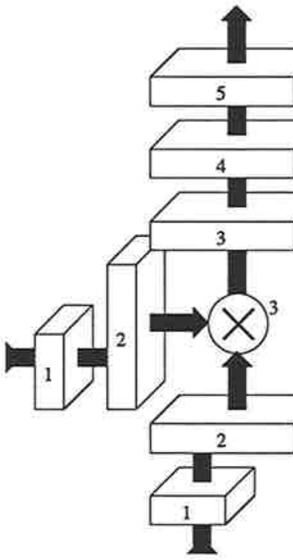


Figure 6.1: Simplified signal flow graph of the convolution algorithm. The numbers within the blocks refer to the stages of the algorithm.

The relation of the algorithmic description and the signal flow graph description is briefly pointed out in figure 6.1. The figure is a schematic interpretation of figure 5.9. Moreover, the overlap-add/overlap-discard operation is also added to the scheme. The numbers of the blocks in figure 6.1 refer to the stages of the algorithmic descriptions. Observe that each block schematically represents a for-loop (see the algorithms 1 and 2 on pages 84 and 85) in which the basic operation is an (I)FFT. In fact, figures 5.9 and 6.1 clearly show the parallel and regular structure of the algorithm.

Based on the signal flow graph we can design several hardware architectures. The design consideration will now be the hardware specifications. For example, if performance is crucial and processor volume is less important, then the several blocks can be implemented in a pipelined architecture. Moreover, within each block the available parallelism, see figure 5.9, can be exploited. On the other hand, if we require compact processor hardware, the hardware can be reused by executing the various FFTs and IFFTs sequentially on it. The processor itself can still be a parallel processor. As we have mentioned in the introduction of this chapter, our final goal is a single chip processor. The basis for the next sections is thus an iterative architecture.

Algorithm 1 : Overlap-Add Convolution Algorithm

```
STAGE 1: for $m = 0 : M_a - 1$
             $\{A_{m,0}, \cdots, A_{m,2M_1-1}\} \overset{FFT}{\longleftarrow} \{a_{mM_1}, \cdots, a_{(m+1)M_1-1}\}$
         end
         for $m = 0 : M_x - 1$
             $\{X_{m,0}, \cdots, X_{m,2M_1-1}\} \overset{FFT}{\longleftarrow} \{x_{mM_1}, \cdots, x_{(m+1)M_1-1}\}$
         end
STAGE 2: for $n = 0 : 2M_1 - 1$
             $\{A'_{0,n}, \cdots, A'_{2M_2-1,n}\} \overset{FFT}{\longleftarrow} \{A_{0,n}, \cdots, A_{M_a-1,n}\}$
         end
         for $n = 0 : 2M_1 - 1$
             $\{X'_{0,n}, \cdots, X'_{2M_2-1,n}\} \overset{FFT}{\longleftarrow} \{X_{0,n}, \cdots, X_{M_x-1,n}\}$
         end
STAGE 3: for $n = 0 : 2M_1 - 1$
             for $m = 0 : 2M_2 - 1$
                 $\widehat{Y}'_{m,n} \leftarrow A'_{m,n} \times X'_{m,n}$
             end
             $\{\widehat{Y}_{0,n}, \cdots, \widehat{Y}_{2M_2-1,n}\} \overset{IFFT}{\longleftarrow} \{\widehat{Y}'_{0,n}, \cdots, \widehat{Y}'_{2M_2-1,n}\}$
         end
STAGE 4: for $m = 0 : M_a + M_x - 2$
             $\{\widehat{y}_{m,0}, \cdots, \widehat{y}_{m,2M_1-1}\} \overset{IFFT}{\longleftarrow} \{\widehat{Y}_{m,0}, \cdots, \widehat{Y}_{m,2M_1-1}\}$
         end
STAGE 5: for $m = 0 : M_a + M_x - 1$
             for $n = 0 : M_1 - 1$
                 $y_{mM_1+n} \longleftarrow \widehat{y}_{m,n} + \widehat{y}_{m-1,M_1+n}$
             end
         end
```

Algorithm 2 : Overlap-Discard Convolution Algorithm

```
STAGE 1: for $m = 0 : M_a - 1$
             $\{A_{m,0}, \cdots, A_{m,2M_1-1}\} \overset{FFT}{\longleftarrow} \{a_{mM_1}, \cdots, a_{(m+1)M_1-1}\}$
         end
         for $m = 0 : M_x$
             $\{\widehat{X}_{m,0}, \cdots, \widehat{X}_{m,2M_1-1}\} \overset{FFT}{\longleftarrow} \{x_{(m-1)M_1)}, \cdots, x_{(m+1)M_1-1}\}$
         end
STAGE 2: for $n = 0 : 2M_1 - 1$
             $\{A'_{0,n}, \cdots, A'_{2M_2-1,n}\} \overset{FFT}{\longleftarrow} \{A_{0,n}, \cdots, A_{M_a-1,n}\}$
         end
         for $n = 0 : 2M_1 - 1$
             $\{\widehat{X}'_{0,n}, \cdots, \widehat{X}'_{2M_2-1,n}\} \overset{FFT}{\longleftarrow} \{\widehat{X}_{0,n}, \cdots, \widehat{X}_{M_x,n}\}$
         end
STAGE 3: for $n = 0 : 2M_1 - 1$
             for $m = 0 : 2M_2 - 1$
                 $\hat{\hat{Y}}'_{m,n} \longleftarrow A'_{m,n} \times \widehat{X}'_{m,n}$
             end
             $\{\hat{\hat{Y}}_{0,n}, \cdots, \hat{\hat{Y}}_{2M_2-1,n}\} \overset{IFFT}{\longleftarrow} \{\hat{\hat{Y}}'_{0,n}, \cdots, \hat{\hat{Y}}'_{2M_2-1,n}\}$
         end
STAGE 4: for $m = 0 : M_a + M_x - 1$
             $\{\hat{\hat{y}}_{m,0}, \cdots, \hat{\hat{y}}_{m,2M_1-1}\} \overset{IFFT}{\longleftarrow} \{\hat{\hat{Y}}_{m,0}, \cdots, \hat{\hat{Y}}_{m,2M_1-1}\}$
         end
STAGE 5: for $m = 0 : M_a + M_x - 1$
             for $n = 0 : M_1 - 1$
                 $y_{mM_1+n} \longleftarrow \hat{\hat{y}}_{m,M_1+n}$
             end
         end
```

6.3 Prototype Architecture

The hardware specification is that the processor must be as small as possible, so we choose the iterative architecture, as it is shown in figure 6.2.a. In figure 6.2.b the data flow per stage is shown. We use off-the-shelf components (FFT processor, multiplier, memory, etc.). We have to exploit the parallel and regular structure rather than hardware technology. At this point, our methodology proves its usefulness, in that, it leads to an efficient mapping of the algorithm onto the available architecture.

Processing parameters

We start with definition of the processing parameters from which we can derive the hardware specifications. The effective data rate D is defined as

$$\mathsf{D} \triangleq \frac{\text{no. of data samples to be processed}}{\text{processing time}}$$

The effective data rate describes, in fact, the *throughput* of a component, whereas the component itself is described as a black-box.[1] The advantage of the definition of the effective data rate is that we do not have to specify a component by its internal specifications (no. of operations, clock frequencies, etc.). I.e., we have shielded the external behavior of a processor from its internal activities, so that we can concentrate on the design of the architecture.

Let us assume that we have an off-the-shelf FFT processor and multiplier, with data rates

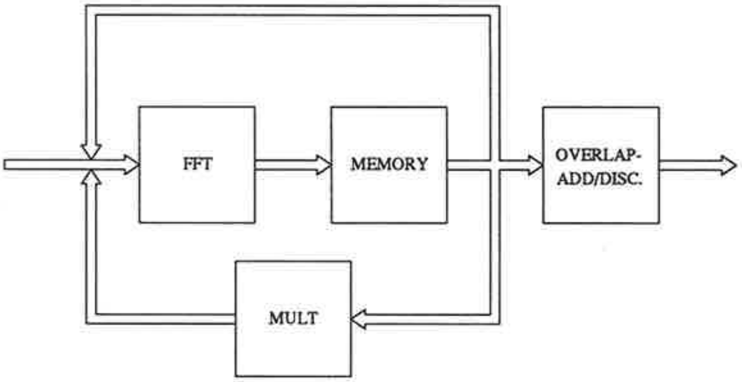
- D_{f1}: the data rate of the FFT processor in $2M_1$-point mode
- D_{f2}: the data rate of the FFT processor in $2M_2$-point mode
- D_m: the data rate of the complex multiplier

To increase the data rate we can place P_f FFT processors and P_m complex multipliers in parallel. Furthermore, we assume that the memory is dual ported, and that I/O speed of the memory is fast enough. To determine the required number of components we introduce the processing time per stage T_i, where i is the stage number. At this moment we can determine T_i in terms of M_a, M_x, M_1 and M_2 and the effective data rates. Furthermore we define a parameter Δ as

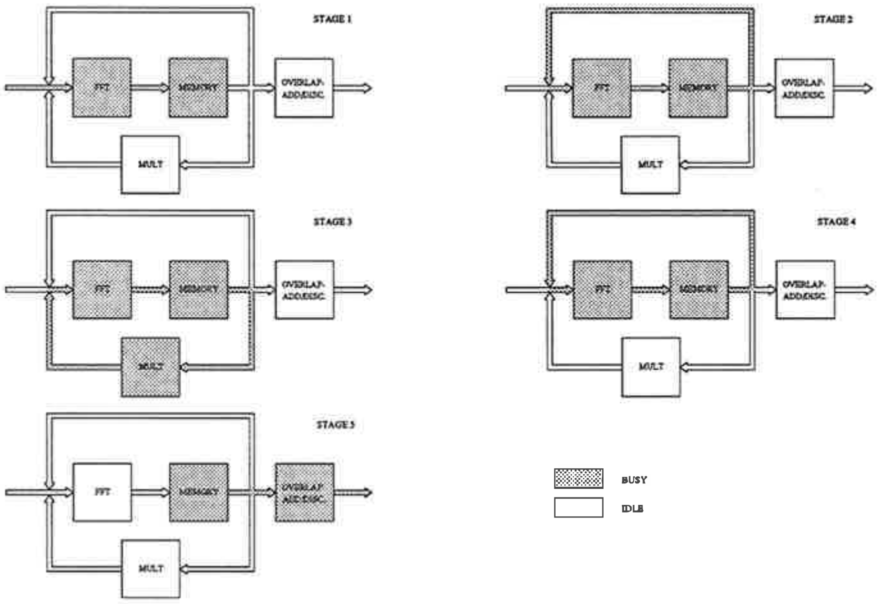
$$\Delta \triangleq \begin{cases} 1 & \text{in case of overlap-discard} \\ 0 & \text{in case of overlap-add} \end{cases}$$

This parameter allows us to express T_i independent of the two algorithms.

[1] Data busses are also considered as components.



(a)



(b)

Figure 6.2: Iterative hardware architecture (a) and the data flow of each stage (b).

STAGE 1: The number of $2M_1$-point FFTs is $M_a + M_x + \Delta$. Thus T_1 is

$$T_1 = \frac{2M_1(M_a + M_x + \Delta)}{P_f \mathsf{D}_{f1}}$$

STAGE 2: The number of $2M_2$-point IFFTs is $4M_1$. Thus T_2 is

$$T_2 = \frac{8M_1M_2}{P_f \mathsf{D}_{f2}}$$

STAGE 3: In stage 3 the processing time is determined by the minimum of the processing speed of the multiplication and the $2M_2$-point IFFT. The number of multiplications is $4M_1M_2$, and the number of $2M_2$-point IFFTs is $2M_1$. Thus T_3 is

$$T_3 = \min\left\{\frac{4M_1M_2}{P_m \mathsf{D}_m}, \frac{4M_1M_2}{P_f \mathsf{D}_{f2}}\right\}$$

STAGE 4: The number of $2M_1$-point IFFTs is $M_a + M_x - 1 + \Delta$. Thus T_4 is

$$T_4 = \frac{2M_1(M_a + M_x - 1 + \Delta)}{P_f \mathsf{D}_{f1}}$$

STAGE 5: The processing time is now determined by the speed of the additions in case of the overlap-add algorithm. In that case we will assume that the additions can be performed at the I/O speed of the memory. In case of the overlap-discard algorithm the data rate is only determined by the I/O speed of the memory. Therefore we will assume that $T_5 \approx 0$ for both algorithms (which is a direct consequence of the assumption that the I/O speed of the memory is no bottleneck.

To determine the overall data rate D of the hardware architecture we assume that the incoming signal to be processed is $\{x_n\}$, i.e. the signal $\{a_n\}$ can be interpreted as an N_a-taps FIR filter. Then

$$\mathsf{D} = \frac{N_x}{\sum_{i=1}^{5} T_i} \tag{6.1}$$

If we assume that $P_m \mathsf{D}_m \geq P_f \mathsf{D}_{fi}$, $i = 1, 2$, (i.e., the multiplication is always faster than the FFT processing) then equation (6.1) becomes

$$\begin{aligned}\mathsf{D} &= \frac{N_x P_f}{2M_1(\mathsf{D}_{f1}^{-1}(2M_a + 2M_x - 1 + 2\Delta) + 6\mathsf{D}_{f2}^{-1}M_2)} \\ &\approx \frac{N_x P_f}{4M_1M_2(2\mathsf{D}_{f1}^{-1} + 3\mathsf{D}_{f2}^{-1})}\end{aligned} \tag{6.2}$$

In the latter step we have used the fact that $2M_2$ is the smallest power of 2 that is greater than $M_a + M_x - 1 + \Delta$. If we assume optimum matching of M_a, M_x and M_2 then $2M_2 \approx M_a + M_x - 1 + \Delta$.

$2M_1$	$2M_2$	D ($\times N_x P_f$)	N_x (maximum)	memory capacity (samples)
16	16	$8 \cdot 10^3$	$128 - N_a$	512
16	64	$2 \cdot 10^3$	$512 - N_a$	$2K$
16	256	$4 \cdot 10^2$	$2K - N_a$	$8K$
16	1K	$1 \cdot 10^2$	$8K - N_a$	$32K$
64	16	$2 \cdot 10^3$	$512 - N_a$	$2K$
64	64	$4 \cdot 10^2$	$2K - N_a$	$8K$
64	256	$1 \cdot 10^2$	$8K - N_a$	$32K$
64	1K	31	$32K - N_a$	$128K$
256	16	$4 \cdot 10^2$	$2K - N_a$	$8K$
256	64	$1 \cdot 10^2$	$8K - N_a$	$32K$
256	256	32	$32K - N_a$	$128K$
256	1K	8	$131K - N_a$	$524K$
1K	16	$1 \cdot 10^2$	$8K - N_a$	$32K$
1K	64	31	$32K - N_a$	$128K$
1K	256	8	$131K - N_a$	$524K$
1K	1K	2	$524K - N_a$	$2M$

Table 6.1: Performance figures and memory requirements for an architecture with a single-chip state-of-the-art FFT processor. M_1, M_2, N_x and the memory capacity are represented by no. of complex samples. The data rate D is represented by the no. of complex samples per second.

Hardware design

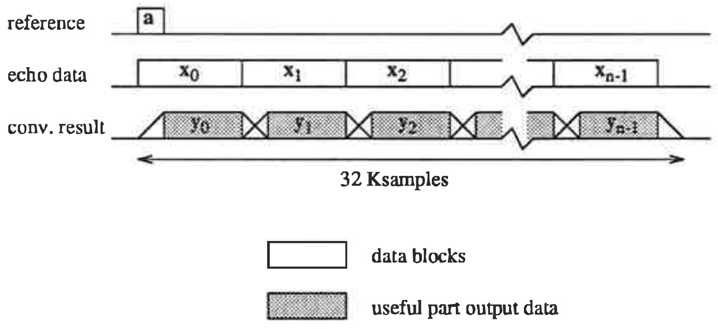
To illustrate the performance of an implementation in hardware with off-the-shelf components we have simulated the architecture using the specifications of a single-chip state-of-the-art FFT processor [GEC93a]. The FFT processor has 4 modes: 16 point, 64 point, 256 point and 1024 point. The data rates for these modes are $\mathsf{D}_{fi} \approx 10$ Msamples/sec, $i = 1, 2$. The performance figures and memory requirement are listed in table 6.1. Observe that the data rate increases linearly with the number of FFT processors in parallel. The number of parallel FFT processors is limited by the I/O data rate $\mathsf{D}_{I/O}$: $P_{fi} \leq \lceil \mathsf{D}_{I/O}/\mathsf{D}_{fi} \rceil$. In our case, the I/O data rate is $\mathsf{D}_{I/O} = 20$ Msamples/sec, thus the maximum number of parallel FFT processors is 2.

Choose for simplicity $M_1 = M_2$, i.e., we use only one FFT mode. Furthermore, the multiplier runs at $\mathsf{D}_m = 20$ Msamples/sec [GEC93b], thus one multiplier is sufficient: $P_m = 1$. Recall that the convolution length is specified at $N_x + N_a + 1 = 8$ Ksamples. Then we have that the FFT processor should operate in 256 point mode. From the table we then read that the maximum data sequence length is $N_x = 32$ Ksamples. Since the actual data sequence length is less than 8 Ksamples, we have that the utilization of the processor (the number of actual samples divided by the number of processed samples) is low: less than 25 %. However, if subsequent data sequences, say $\mathbf{x}_0, \cdots, \mathbf{x}_{n-1}$, are convolved with a fixed reference sequence, say $\mathbf{a}$, then we can construct a long data sequence $\mathbf{x} = [\mathbf{x}_0 \cdots \mathbf{x}_{n-1}]$ and convolve it with $\mathbf{a}$. The adjacent convolution results $\mathbf{y}_i = \mathbf{a} * \mathbf{x}_i$ have an overlap of N_a samples. However, in general we are not interested in the first and last $N_a - 1$ samples of $\mathbf{y}_i$. The principle is shown in figure 6.3.a. If the length $\mathbf{x}$ approaches 32 K $- N_a$, then the utilization of the processor approaches 100 %.

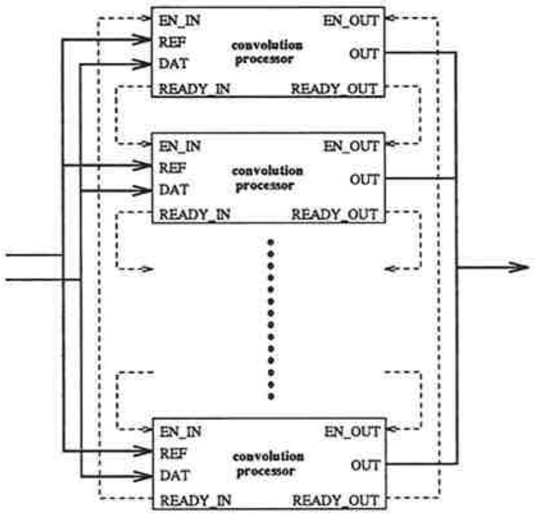
From table 6.1 we obtain the effective data rate of the convolution processor $\mathsf{D} = 32 N_x P_f =$ 2.0 Msamples/sec. Thus we do not reach the specified data rate of 10 Msamples/sec. However, a simple mechanism is used to increase the data rate by putting multiple convolution processors in parallel with negligible additional overhead. Each convolution processor has an EN_IN and EN_OUT input and a READY_IN and READY_OUT output, see figure 6.3.b. The convolution processor has the ability to enable its input buses REF and DAT. If EN_IN is set, the convolution processor enables REF and DAT. Initially, one of the convolution processor has enabled input buses REF and DAT, and the other convolution processors have disabled input buses REF and DAT. When the input data has been read into convolution processor, it disables its input buses REF and DAT and generate a READY_IN signal, which is the EN_IN of next convolution processor. This mechanism can be readily applied to the output bus OUT. A convolution processor can only write its convolution result if its output bus OUT is enabled by an EN_OUT signal. If the convolution result has been written, its output bus OUT is disabled and a READ_OUT signal is generated, which is the EN_OUT of the next convolution processor. The next convolution processor can then write its convolution result to the output bus. This simple mechanism allows us to consider the array of convolution processors as a single convolution processor, with an effective data rate proportional to the number of convolution processors within the array.

Evaluation

In general, off-the-shelf FFT processors use block-floating point data format. During the design of the prototype architecture it appeared to be a major problem to implement the block-exponent administration in hardware. It requires typically combined control and data path logic. We have implemented this in Programmable Logic Devices (PLDs) [Alt93] for they allow a high degree of design flexibility. However, it appears that mapping of combined control and data path functions onto multiple PLDs, results in much data communication between PLDs. These aspects made it extremely difficult to manage the design trajectory of the prototype architecture. Moreover, due to the block-floating point data format, the communication busses are wide (in our case 40 bits), and obviously, the required working memory is large. The conclusion is that an implementation of the convolution processor using off-the-shelf DSP components is not fully satisfying, especially if it must be fast and small and have low power consumption, and if it must also meet stringent dynamic range and accuracy specifications.



(a)



(b)

Figure 6.3: In (a) the simplified timing diagram of the convolution of the subsequent data sequences $\mathbf{x}_i$, $i = 0, \cdots, n-1$, with the reference sequence **a** is shown. Since we are only interested in the body samples of the results $\mathbf{y}_i$, the overlaps can be discarded. In (b) a convolution processor array is shown. REF and DAT represent input data buses for the reference data and the echo data, respectively, and OUT represents the output data bus for the convolution results. The READY_IN and EN_IN signals are used to enable/disable the input data buses and the READY_OUT and EN_OUT signals are used to enable/disable the output data buses.

6.4 Dedicated VLSI Architecture

In this section we present a dedicated VLSI architecture for the convolution processor. The specifications are mainly based on the prototype architecture described in previous section, in that, the chip has 2 Msamples/sec effective data rate and supports parallel processing without additional overhead. However, in the prototype architecture we were forced to use a block-floating point data format (due to the format used in the FFT processor). If we stick to this data format, we definitely meet the same problems as in the prototype architecture design trajectory: much overhead due to block-exponent administration and large word widths. Therefore, we propose a hybrid floating point data format[2] for the VLSI architecture. The basic processing operation is now a 128-point FFT, thus a maximum convolution length of 8 Ksamples is allowed. The 128-point FFT is realized in radix 2 processor elements for it has a simple structure. Therefore, we put our main effort in the following items:

- designing parallel structures for multiple radix 2 processors on chip
- determination of trade-off between accuracy and dynamic range of the I/O at the one hand and the internal arithmetic accuracy at the other hand

The proposed architecture, including the finite precision arithmetic operations, is modelled in MatLab, a high level programming tool for numerical analysis. The functionality and the accuracy are evaluated by simulations, using several representative data sets, including real SAR data. The feasibility of the architecture in terms of chip area is validated based on standard cell libraries of the ASIC design tool COMPASS [COM94].

Global architecture description

The global structure of the hardware architecture is shown in figure 6.4. Its main part is a 128 point FFT processing core, containing a fully parallelized radix 2 processor array as well as the FFT working memory. Input and output registers take care of I/O buffering to guarantee a continuous I/O throughput. The I/O data rate and the effective FFT processing data rate are both 20 Msamples/sec., such that optimum balancing between I/O data rate and FFT processing data rate is achieved. The size of the convolution working memory is 16 Ksamples, which cannot yet be implemented efficiently into on-chip RAM using standard cell. However, since the I/O load of the convolution working memory is not the limiting factor, it can be implemented in external RAM without affecting the effective convolution data rate.

[2]A hybrid floating point number consists of three parts: mantissa for the real part, mantissa for the imaginary part, and one exponent.

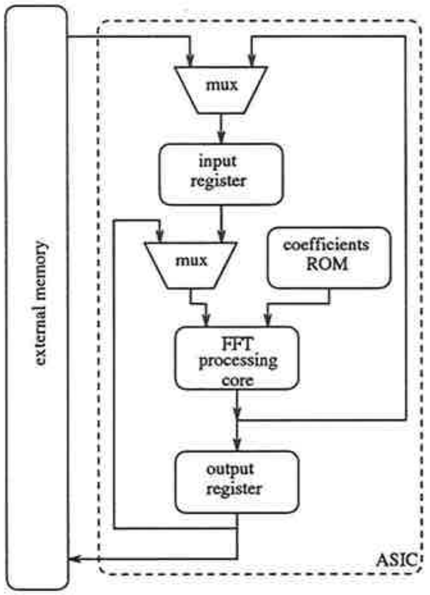


Figure 6.4: The global structure of the hardware architecture. The architecture consists of an external memory and a chip core, including the FFT processing core.

The FFT processing core

The basic operation of our FFT processing core is an FFT stage operation based on radix 2. An optimum FFT stage operation requires high speed dual ported memory accesses, which is often a bottleneck in the design of FFT processing hardware. Due to this fact increasing the radix 2 performance will not lead automatically to a proportional increment of the FFT stage performance. To overcome this problem we propose a chip implementation of the FFT algorithm based on the FFT algorithm proposed by Singleton [Sin67]. This algorithm has the same data path for each FFT stage operation. The purpose was to efficiently perform large FFTs using a dedicated radix 2 processor with limited amount of RAM. The latter problem was solved by using external sequential data storage (magnetic tapes or disk files). The proposed configuration consists of a single radix 2 processor and 4 external input files and 4 external output files. Nowadays in high speed FFT processing memory access time is often the bottleneck rather than a limited amount of RAM. The algorithm proposed by Singleton uses separate memory units allowing us to perform multiple memory accesses in parallel, which decrease the speed requirements of the memory units. The same principle holds for a VLSI implementation of the radix 2 processor including the sequential data storage, where the latter can be implemented

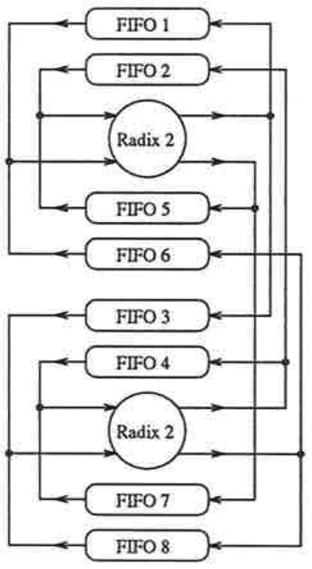


Figure 6.5: The FFT processor including the 2 radix 2 processing cores which achieve full parallelism.

efficiently using 4 FIFOs rather than 8 sequential register files. Moreover, in [Sto71] it was shown that the structure of this algorithm can be mapped onto a fully parallel radix 2 processor array, where the processing speed of the FFT stage operation depends linearly on the number of radix 2 processors. The number of FIFOs remains 4 per radix 2 processor. The memory access (i.e. the FIFO I/O handling) is realized using wiring rather than addressing resulting in optimum utilization of the parallel radix 2 processors. In figure 6.5 an example is given of the hardware architecture of our proposed parallel radix 2 processor array using 2 radix 2 processing cores and 8 FIFOs.

The radix 2 processor

In this section we will propose the architecture of the radix 2 processing core that will be used in our parallel radix 2 processor array. Its data path is based on a hybrid floating point data format that consists of two mantissas (real and imaginary) and a single exponent. In many DSP applications, and especially in SAR processing, dynamic range in combination with preservation of accuracy is essential. We will show that our floating point data format behaves as optimally as full floating point data formats in terms of dynamic range and accuracy. However, a major

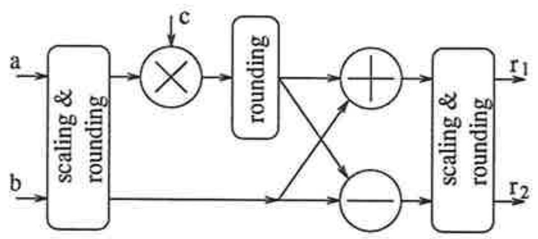


Figure 6.6: The architecture of the radix 2 processor. The exponent is bypassed from input scaling & rounding block to output scaling & rounding block.

advantage above full floating point data format is that it can be implemented using small, simple and efficient fixed point arithmetic hardware structures with little scaling overhead.

The proposed radix 2 processing core architecture is shown in figure 6.6. The arithmetic operations are assumed fixed point, whereas the scaling & rounding is assumed part of the data path. Inputs a and b represent the two data inputs of the radix 2 and input c represents the coefficient input. The exponent widths of a and b are equal and sufficiently large. Furthermore the coefficient c has no exponent since the coefficients are all part of the unit circle, i.e. their dynamic range is constant. Observe that for this reason the multiplication result of a and c has the same exponent as the exponent of a. Therefore the exponent handling which should be required for the floating point arithmetic operations can be relocated to the input of the radix 2 processing core.

After a fixed point operation such as scaling and multiplication the word width becomes substantially larger. At first sight this implies an increment in accuracy which obviously requires large arithmetic hardware. However, this increment in accuracy is only virtual in that the number of significant bits of the output of a fixed point operation is always less or equal than the minimum number of significant bits of the inputs of the fixed point operation. This implies that the word width at the output of the fixed point operation can be rounded without loss of accuracy, as we will show now. The starting point is a concise error analysis based on the simplified data flow shown in figure 6.7. The samples are assumed real but the extension to the complex case is straight forward. Furthermore we assume that the inputs a and b and the coefficient c are stochastically independent. The fact that the actual coefficients c are deterministic does not affect this analysis. Let a, b and c be wide sense stationary with zero mean and let them be uniformly distributed between -1 and 1. Then the expected signal power is given as $\mathsf{E}a^2 = \mathsf{E}b^2 = \mathsf{E}c^2 = 1/3$, with E the expectation operator. Assume that a, b and c are represented in m, n and p bits, respectively. Let e_a, e_b and e_c be uniformly distributed error noise in a, b and c, respectively, then $|e_a| \leq 2^{-m}$, $|e_b| \leq 2^{-n}$ and $|e_c| \leq 2^{-p}$. Then the expected error power of a, b and c is $\mathsf{E}e_a^2 = (1/3)2^{-2m}$, $\mathsf{E}e_a^2 = (1/3)2^{-2n}$ and $\mathsf{E}e_a^2 = (1/3)2^{-2p}$, respectively.

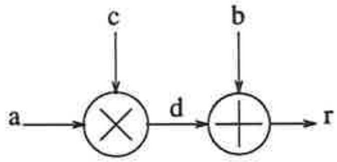


Figure 6.7: The simplified model of the data flow in the radix 2 processor with inputs a, b, coefficient c, intermediate result d and output r.

Thus, after the multiplication the error-to-signal ratio is

$$\mathsf{E}e_d^2/\mathsf{E}d^2 = \mathsf{E}e_a^2/\mathsf{E}a^2 + \mathsf{E}e_c^2/\mathsf{E}c^2 = 2^{-2m} + 2^{-2p} \tag{6.3}$$

The expected error power of the output r after the addition is:

$$\begin{aligned} \mathsf{E}e_r^2 &= \mathsf{E}e_d^2 + \mathsf{E}e_b^2 = 1/9\left(2^{-2m} + 2^{-2p}\right) + (1/3)2^{-2n} \\ &\approx (1/3)\left(2^{-2(m+0.8)} + 2^{-2(p+0.8)} + 2^{-2n}\right) \end{aligned} \tag{6.4}$$

From equation (6.4) follows that the number of significant bits in the output will never be more than the minimum value of $\min\{m + 0.8, n, p + 0.8\}$, which is the minimum number of bits required to represent the output r. Conversely we can state that it has no use to represent the inputs with different accuracy, so the number of bits to represent the different inputs and the output must be the same, i.e., $m = n = p$. The inputs a and b are scaled to a single exponent. From equation (6.4) it follows that the output errors in r_1 and r_2 are determined by the largest input error in a, b and c, implying that a and b can be rounded to the original word width after scaling. Furthermore the multiplication result will never have more significant bits than the number of bits of the least significant input. Since we have stated before that all bits in the coefficients are significant we know that all significant bits will be in the upper half of the result. This implies that we are also allowed to round after the multiplication. In fact also after addition the result can be rounded. However, the increment after an addition/subtraction is 1 bit, so that the additional rounding hardware is relatively expensive with respect to the increment. Moreover, depending on the appearance of an overflow bit, either the result should be rounded or the overflow bit should be eliminated. This operation is performed at the output of the radix 2 processor by the scaling & rounding unit.

Simulation results

The parallel radix 2 processor array of figure 6.5 (including the radix 2 processing core architecture of figure 6.6) is simulated to verify the results of the error analysis. The data formats were specified on 12 bits real and imaginary mantissa width for both data and coefficient samples and

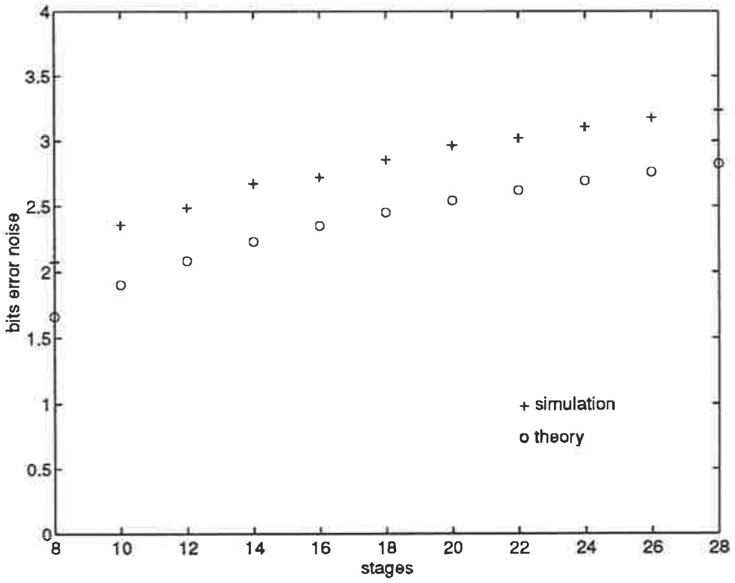


Figure 6.8: The number of error bits plotted versus the number of FFT stages. The simulation results contain approximately 0.4 bits more error noise than the theoretical values.

8 bits exponent width for the data samples. After the complex multiplication (see figure 6.6) the result is rounded to 13 bits (a complex multiplication includes an addition, introducing an overflow bit). After the addition/subtraction the results are scaled and rounded from 14 bits to 12 bits, which was the original data format.

The accuracy of the proposed hardware architecture has been simulated for several FFT lengths. It has been compared with the accuracy for full floating point FFT hardware, which was obtained in [Wei69]. The simulation results are shown in figure 6.8. It has been found that, indepently of the FFT length, our FFT processor hardware is 0.4 bits less accurate than the full floating point FFT hardware. For our application (SAR processing) a convolution length of 8K is required, implying 128 points FFTs [Boe95]. Each data sample passes 28 (I)FFT stages, which results in a loss of accuracy of 3.2 bits, see figure 6.8. Thus our 12 bits floating point FFT architecture has 8 bits accuracy. The FFT processor hardware was also simulated as a part of the fast convolution hardware architecture, see figure 6.4. Two representative input data sets has been fed into the simulated architecture, see figures 6.9 and 6.10. It was found that the output accuracy has about 8 bits accuracy in all cases, which verified our expectations. For completeness we give a brief notification. The relocation of the rounding behind the multiplication to the output (see figure 6.6) will improve the overall accuracy with $\mathcal{O}(10^{-1})$ bits. However, in terms of additional hardware it will cost approximately an extra complex adder.

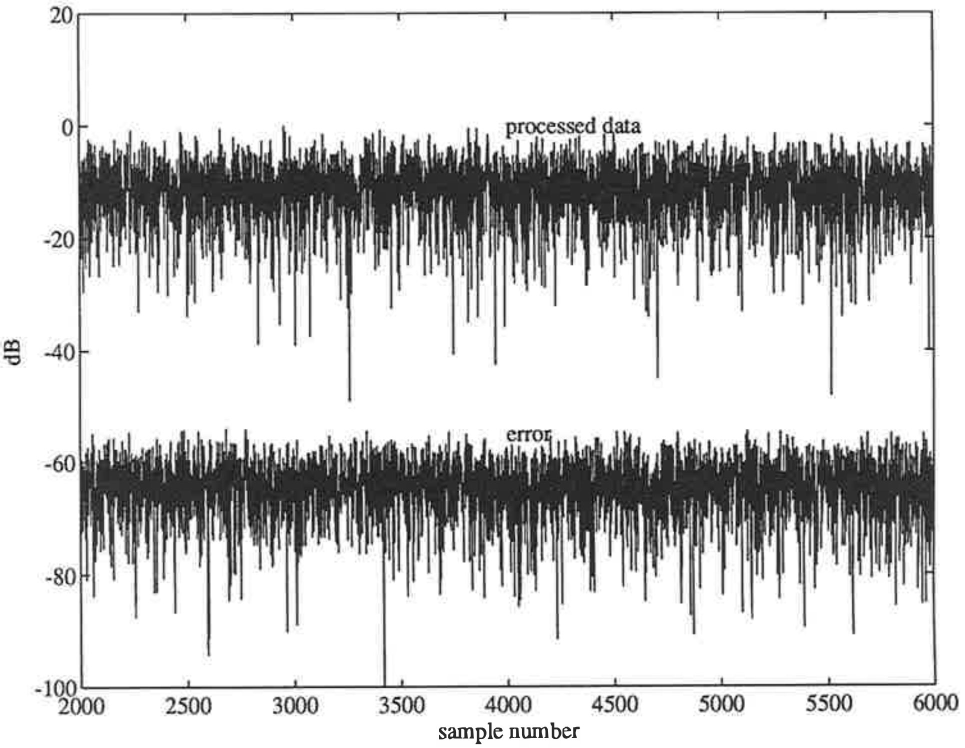


Figure 6.9: Simulation of the convolution architecture using random data. The accuracy of the input data is 8 bits, the internal accuracy of the architecture is 12 bits. The error in the output data is approx. 50 dB below the convolution result, implying a convolution accuracy of approx. 8 bits. The error is obtained by comparing the amplitude of the simulated convolution result with the amplitude of full floating point convolution result computed with MatLab.

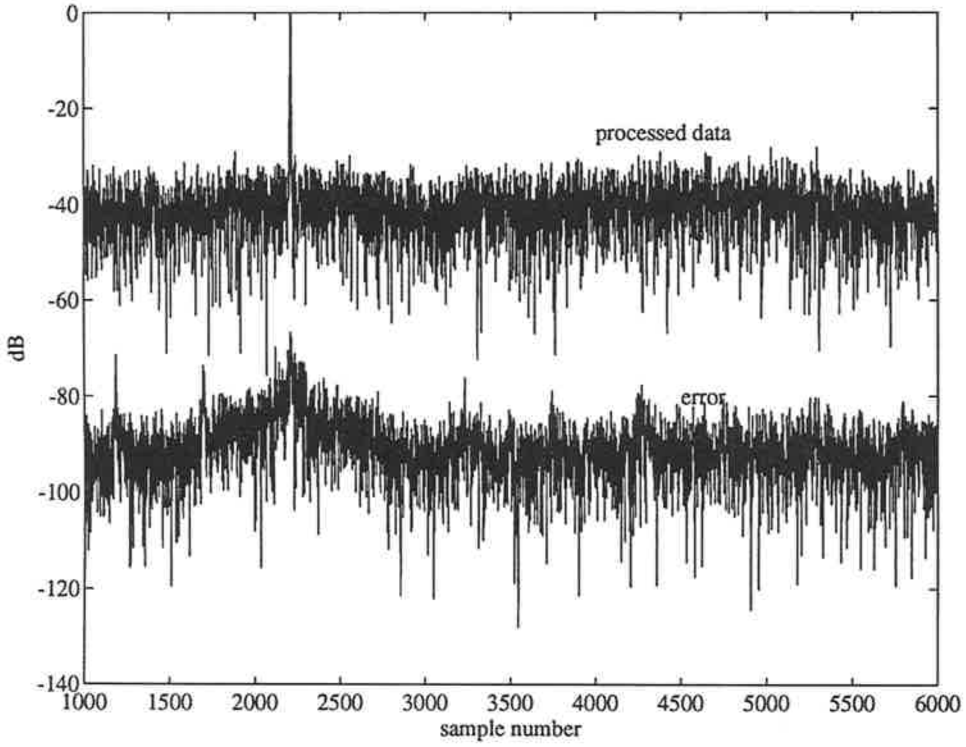


Figure 6.10: Simulation of the convolution architecture using real SAR data. The data is part of an scene with crop fields in which corner reflecters are placed (the peak in the data). The accuracy of the input data is 8 bits, the internal accuracy of the architecture is 12 bits. The error in the output data is approx. 50 dB below the convolution result, implying a convolution accuracy of approx. 8 bits. It is clearly shown that despite the increase in dynamic range due to the appearance of the peak within the convolution result, the accuracy remains 8 bits. This is due to the use of floating point data format.

6.5 Concluding Remarks

In this chapter we have presented the design of a hardware VLSI architectures of the multirate convolution system. It resulted from the mapping of the graphical representation of the multirate convolution systems obtained in chapters 4 and 5. Two hardware implementations are proposed:

A prototype architecture: A prototype architecture is proposed, to handle typical hardware design problems, such as finding a trade-off between I/O data rate, processing data rate, flexibility, performance, modularity and parallelization. It is based on off-the-shelf DSP components (FFT processor and complex multiplier [GEC93a, GEC93b]) and Programmable Logic Devices (PLDs) [Alt93]. It resulted in a hardware architecture design which is realized at the moment of writing.

A VLSI architecture: Based on the hands-on experience gained in the design of the prototype architecture, we have presented a hardware VLSI architecture. The feasibility in terms of accuracy is validated by simulations of a software model of the VLSI architecture. Hereby we have used real SAR data. Based on standard cell libraries (0.6 μ) of the ASIC design tool COMPASS [COM94], the feasibility of a single-chip implementation is validated in [Boe95].

Implementation of the multirate convolution system in a VLSI architecture allows more design freedom in terms of I/O and FFT data rate balance, accuracy and dynamic range compared to an implementation in off-the-shelf DSP components. We have focused on the efficient implementation and the arithmetic accuracy of the convolution processor. The result is a floating point convolution processor architecture including a parallel FFT processor core based on the FFT algorithm proposed by Singleton [Sin67]. Although this algorithm was developed three decades ago, its efficiency in terms of memory management and data flow management have gained renewed interest [BCM94].

A remarkable side-effect is that our VLSI implementation requires twice as much computations compared to a straight-forward implementation in DSP boards. Nevertheless, it performs much better than a DSP board solution in terms of processing speed, volume and power consumption[3] and, in many cases, also in terms of accuracy and dynamic range.

[3]Although in some cases a single DSP board has higher processing speed, the chip has higher processing speed *per units of hardware size and per units of power concumption.* For example, a good measure would be the processing speed per m^2 of printed circuit board area and per Watt power concumption.

CHAPTER 7

CONCLUDING REMARKS

In this thesis we have introduced a schematic design methodology for designing efficient hardware architectures for multirate convolutions. Other multirate signal processing systems can be similarly designed. The design method uses graphical representations of algebraic substitutions and equivalencies. Graph manipulation is much simpler than equation manipulation, in particular when the equations are written out in full low-level details. It is known that multirate filter banks and fast convolution schemes have a close relationship. Using the graph manipulation techniques that we have developed, this relationship has been made explicit in a clarifying and illustrative manner. The design methodology allows direct mapping of the multirate convolution algorithm onto prototype architectures or dedicated VLSI processors.

In many complex real-time signal processing applications, such as radar, sonar and medical imaging, a schematic design methodology is desirable in which typical application specific constraints such as processing speed, processor size and memory utilization can easily be taken into account. The nature of our design methodology is that it partitions a complex signal processing algorithm into small, less complex sub-algorithms. It leads to a highly structured design trajectory of algorithms and architectures for complex real-time signal processing applications. In the end, this approach leads to an effective gain in design time and reduces design risks and complexity.

We have demonstrated the usefulness of the methodology by designing a multirate convolution processor which is applied in range and azimuth compression for real-time SAR processing. Both range and azimuth compression are being considered as the critical processing steps for such systems, especially when processor size is limited and low power consumption and high processing speed are required. Applying our design methodology has resulted in an efficient VLSI convolution processor. It is expected that a real-time on-board SAR processor based on this convolution processor is up to a factor 10 smaller (in volume and power consumption) than a SAR processor based on commercially available DSP boards, e.g. [Cat92, Mer95].

If we express the performance of a convolution processor in terms of processing speed per unit of processor volume and per unit of power consumption, then the performance of the VLSI convolution architecture outperforms a straight-forward DSP board implementation. A rough estimate is that the processing speed of our convolution chip compares to one state-of-the-art DSP board. However, it is readily seen that volume and power consumption are substantially smaller. Nevertheless, in terms of computational complexity, the VLSI architecture is approximately twice as expensive. This should not be a surprise for hardware designers. One might save a few computations in an algorithm but, in general, this will corrupt the regular structure of the implementation in hardware architecture. In our case, this means that the trade-off between paralellization, control, I/O data rate and memory management would be lost. Moreover, we have shown that our VLSI architecture meets the stringent specifications of radar signal processing on accuracy and dynamic range by using a hybrid floating point format. It is difficult to meet these specifications with a DSP board solution, due to the block-floating point format that is often used.

This gives reason to state that designers of hardware architectures should not express the costs of a DSP architecture in terms of computational complexity, but in terms of *overall performance*. The latter may include for example, effective data rate, processor volume, power consumption, accuracy, and dynamic range. Of course, the exact definition of the overall performance depends on the application, in that it does not make sense to implement, for example, a SAR processor in dedicated VLSI hardware when off-line processing is required. However, in chapter 1 we have listed a number of on-board SAR processing applications for small platforms, such as fighters and unmanned airborne vehicles. In the near future these applications will come within reach, when dedicated VLSI architectures (and especially the convolution architecture) are used. This list can be extended with, for example, applications for on-board satellite SAR processing.

Implementation of the results

At this moment the results of the work presented in this thesis are applied in a number of related projects at TNO Physics and Electronics Laboratory. The realization of the prototype architecture presented in section 6.3 is almost finalized. We estimate that the processor will fit onto a single Euro 6 Printed Circuit Board, and will show less than 20 W of power consumption. The processor will be part of a real-time on-board SAR processor for the PHARUS system. The first version of the SAR processor (the so-called "Real-Time SAR Testbed" [vHDvB$^+$94]) will be operational in early '96.

The realization of the VLSI convolution processor specified in section 6.4 will start in late '95. The VLSI convolution processor will be especially applicable for miniaturized real-time SAR processor architectures. Moreover, the VLSI convolution processor can be applied directly in many other DSP applications that require small, low power and high performance

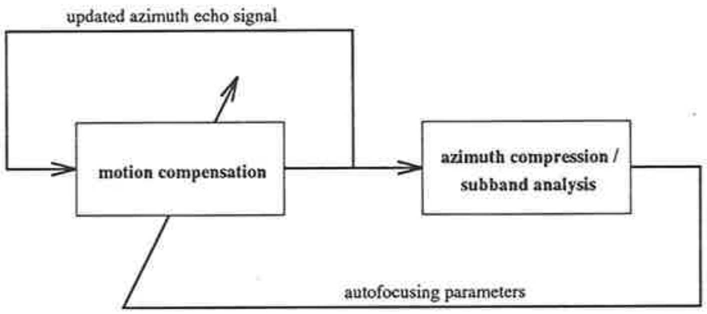

updated azimuth echo signal
motion compensation
azimuth compression / subband analysis
autofocusing parameters

Figure 7.1: Schematic block diagram of iterative autofocusing including an integrated azimuth compression/subband analysis system.

convolvers. We hope that it will prove the feasibility of a real-time SAR processor on-board small platforms, such as small aircrafts, UAVs and satellites. In the latter case, the Netherlands Agency for Aeronautics has commissioned TNO Physics and Electronics Laboratory to carry out a feasibility study of a real-time on-board "focused" satellite SAR processor, which will be based on the presented design methodology.

Future investigations

In this thesis we have not addressed the implications of motion errors on the real-time on-board SAR processor specifications. In practice, we must compensate for the motion errors induced by, for example, turbulence and drift. Motion parameters can be obtained from accurate GPS or INS position and motion measurements, or they can be estimated directly from the echo data using autofocusing techniques, see appendix C. The mapdrift autofocusing method has a close relationship to subband filtering, which is commonly used in multirate filter banks. For example, the multiple looks required for the mapdrift autofocusing can be obtained by bandpass filtering the compressed azimuth echo signal. If these bandpass filters divide the frequency band uniformly (or at least a part of it), we have an analysis filter bank as is used in multirate filter banks. The multiple look images can then be interpreted as channel signals, which was also observed in [CPR89]. It would be desirable to perform an efficient autofocusing scheme on the channel signals, eventually running on reduced data rates. Since we used a multirate convolution system to do the azimuth compression, we can possibly integrate it with the analysis of the subband channels.

In appendix C we have also shown that mapdrift autofocusing in general requires some iteration steps. If we embed these iteration steps in an integrated azimuth compression/subband analysis, we recognize the similarity with adaptive filter schemes, see figure 7.1. The engineering of adaptive filters might then be applicable in our scheme. For example, in [GV92] adaptive filtering in subbands is proposed, which is computationally efficient due to decimation of the channel signals. Moreover, in [ES92] block-adaptive filtering schemes are proposed with efficient use of short-time FFTs. Clearly, these schemes have much in common with our iterative autofocusing scheme, and it should be investigated how we can exploit these techniques in SAR.

APPENDIX A

A MATRIX ALGEBRAIC VIEWPOINT OF SAR

In this appendix we introduce matrix algebra in the SAR processing. For many years SAR data has been analyzed using traditional radar signal processing terminology, such as Doppler shifts, pulse waveforms, and the like. This kind of analysis has been useful for some reasons:

- It illustrates the historical development of SAR processing, see for example section 2.4.
- It explains the several steps of the multi-dimensional SAR processing using analogies in the physical world, such as Doppler shifts, antenna beamwidth and focusing.
- It allows simplifications of the SAR formulations, so that the SAR reconstruction system approximates a set of linear shift-invariant filters (the matched filters).

We have addressed these points extensively in chapter 2, and so did most of SAR experts in the past, e.g., [Sko70, Hov80].

The matrix algebraic viewpoint is relatively new in the field of SAR (as it is in many signal processing applications). The fact that data amounts are so huge makes one sceptic to the usefulness of matrices in SAR, because their dimensions must be gigantic. Indeed, these dimensions are gigantic, typically $\mathcal{O}(10^9 \times 10^9)$, which make the matrices useless in the processing itself. However, they can be extremely useful in analysis of the SAR data, and provide us with novel SAR processing *approaches*. Since it goes beyond the scope of the research project, we do not fully elaborate all relevant matrix algebra.

The acquisition matrix

Here we give a description of the SAR acquisition system impulse response determined in chapter 2. Recall the acquisition system impulse response of equation (2.29) , and consider the following preliminaries:

1. We use the substitution $(t', t) \leftarrow (2r/c, x/v)$, so that the impulse response is function of the spatial variables (r, x) rather than of the temporal variables (t', t).

2. In chapter 2 we assumed that R_0 was constant, whereas here it is considered variable. This aspect makes the impulse response shift-variant for shifts R_0 in r. To denote this shift-variance we add R_0 to the list of variables (if applicable), separated by a semi-colon, and we omit the subscript $_0$.

3. The imaged surface is constituted by a set of points, described by their nominal range R, and their x-coordinate X. Define the reflectivity function $\rho(R, X)$, $0 \leq \rho(R, X) \leq 1$ that assigns to each point on the imaged surface an intensity value. The SAR reconstruction system estimates the reflectivity function.

4. In the remainder of this chapter we do not annotate the dimensions of matrices, vectors, etc.. We simply assume that they all have appropriate dimensions.

Let $s(r, x; R)$ denote the 2D impulse response of the SAR acquisition system, and let $e(r, x)$ denote the received echo signal[1], then their relationship is described as

$$e(r, x) = \int\int s(r - R, x - X; R)\rho(R, X)\,\mathrm{d}R\,\mathrm{d}X \tag{A.1}$$

From equation (2.29) we have

$$s(r - R, x - X; R) = p(r - R - \Delta R(x - X; R))\exp(-j4\pi\Delta R(x - X; R)/\lambda) \tag{A.2}$$

In this chapter we simplify the notation, using

$$b(r - R, x - X; R) \triangleq p(r - R - \Delta R(x - X; R)) \tag{A.3}$$

$$c(x - X; R) \triangleq \exp(-j4\pi\Delta R(x - X; R)/\lambda) \tag{A.4}$$

Then equation (A.1) becomes

$$e(r, x) = \int\int b(r - R, x - X; R)c(x - X; R)\rho(R, X)\,\mathrm{d}R\,\mathrm{d}X \tag{A.5}$$

The discrete version of equation (A.5) can be obtained in two steps. The first step is to sample $e(r, x)$ in range and azimuth, thereby assuming that the sampling criteria in range and

[1]Note that in section 2.4 the echo signal $e(t', t)$ was the impulse response.

azimuth hold. Let r_i and x_j be the discrete spatial instants in range and azimuth, respectively, and let $e_{i,j}$ be the $(i,j)^{th}$ component of the discrete echo signal, then

$$e_{i,j} \triangleq e(r_i, x_j) = \int\int b(r_i - R, x_j - X; R)c(x_j - X; R)\rho(R, X)\,\mathrm{d}R\mathrm{d}X \tag{A.6}$$

In fact, we already have introduced $e_{i,j}$ in section 3.1.

The next step is to consider the reflectivity function only at the uniformly spaced discrete points (R_m, X_n). Let dR and dX be the distance between adjacent points (R_m, X_n) in range and azimuth, respectively, and let dr and dx be the spatial sample distance of $e_{i,j}$ in range and azimuth, respectively. We assume that the following conditions hold: $dR \leq dr$ and $dX \leq dx$. To simplify our analysis, however, we restrict us to the case that $dR = dr$ and $dX = dx$. Moreover, we assume that $r_i \equiv R_m$ and $x_j \equiv X_n$ if $i = m$ and $j = n$, respectively. These assumptions do not affect the generality of our analysis. Thus we can write

$$b(r_i - R_m, x_j - X_n; R_m) = b(r_{i-m}, x_{j-n}; R_m) \triangleq b_{i-m,j-n;m} \tag{A.7}$$

$$c(x_j - X_m; R_m) = c(x_{j-m}; R_m) \triangleq c_{j-m;m} \tag{A.8}$$

Let $\rho_{m,n} \triangleq \rho(R_m, X_n)$, then equation (A.6) becomes

$$e_{i,j} = \sum_m \sum_n b_{i-m,j-n;m}\, c_{j-n;m}\, \rho_{m,n} \tag{A.9}$$

Equation (A.9) can be written in matrix notation. Let the echo vector $\mathbf{e}$ and the reflectivity vector $\boldsymbol{\rho}$ be defined as

$$\begin{aligned}\mathbf{e} &= [\cdots\ e_{i,j}\ e_{i+1,j}\ \cdots\ e_{i,j+1}\ e_{i+1,j+1}\ \cdots] \\ \boldsymbol{\rho} &= [\cdots\ \rho_{m,n}\ \rho_{m+1,n}\ \cdots\ \rho_{m,n+1}\ \rho_{m+1,n+1}\ \cdots]\end{aligned}$$

Then we can write equation (A.9) in matrix notation [BD92]

$$\mathbf{e} = \boldsymbol{\rho}\mathbf{S} \tag{A.10}$$

$\mathbf{S}$ is referred to as the *acquisition matrix*. Observe that the terms $b_{i-m,j-n;m}$ and $c_{j-n;m}$ refer to range and azimuth. We can also construct the banded matrix $\mathbf{B}_{j-n}$ and the diagonal matrix $\mathbf{C}$ related to range and azimuth, respectively, as follows

$$\mathbf{B}_{j-n} \triangleq \left[\begin{array}{cccc} & & & 0 \\ \ddots & \vdots & & \\ \ddots & b_{1,j-n;m-1} & \vdots & \\ & b_{0,j-n;m} & b_{1,j-n;m} & \\ & & b_{0,j-n;m+1} & \ddots \\ 0 & & & \ddots \end{array}\right] \begin{array}{l} \\ \\ \\ \leftarrow m^{th}\ row \\ \\ \\ \end{array} \tag{A.11}$$

$$\mathbf{C}_{j-n} \triangleq \mathrm{diag}\{\cdots, \underset{\substack{\uparrow \\ (m,m)^{th}\ entry}}{c_{j-n;m}}, c_{j-n;m+1}, \cdots\} \tag{A.12}$$

The acquisition matrix $\mathbf{S}$ can then be written as

$$\mathbf{S} \;=\; \begin{bmatrix} \ddots & \vdots & & 0 \\ \ddots & \mathbf{C}_1\mathbf{B}_1 & \vdots & \\ & \mathbf{C}_0\mathbf{B}_0 & \mathbf{C}_1\mathbf{B}_1 & \\ & & \mathbf{C}_0\mathbf{B}_0 & \ddots \\ 0 & & & \ddots \end{bmatrix} \tag{A.13}$$

As we can see now, the acquisition matrix $\mathbf{S}$ is, in fact, a block-convolution matrix, and each block is itself a convolution matrix (albeit somewhat distorted). The matrix $\mathbf{S}$ thus approximates a 2D convolution.

The reconstruction matrix

As we have mentioned, ρ contains the discrete reflectivity function, and is thus the unknown signal to be estimated. Let $\mathbf{H}$ be a reconstruction matrix, then $\widehat{\rho} = \mathbf{e}\,\mathbf{H}$ is an estimate of ρ. We will now describe the SAR reconstruction in terms of matrix algebra. In deriving a reconstruction matrix $\mathbf{H}$, we anticipate to practical constraints (e.g., the dimension of the matrix $\mathbf{S}$ and the complexity of matrix algorithms). The SAR reconstruction matrix that we consider is based on matched filtering the data, see section 2.4. If we neglect the range migration then $\mathbf{B}_{j-n} = \mathbf{B}_0$ for all $j-n$. Observe that $\mathbf{B}_0$ is a convolution matrix. Thus $\mathbf{S}$ can be approximated as

$$\mathbf{S} \;=\; \begin{bmatrix} \ddots & & 0 \\ & \mathbf{B}_0 & \\ & & \mathbf{B}_0 \\ 0 & & \ddots \end{bmatrix} \begin{bmatrix} \ddots & \vdots & & 0 \\ \ddots & \mathbf{C}_1 & \vdots & \\ & \mathbf{C}_0 & \mathbf{C}_1 & \\ & & \mathbf{C}_0 & \ddots \\ 0 & & & \ddots \end{bmatrix} \triangleq \mathbf{B}\,\mathbf{C} \tag{A.14}$$

There exist two permutation matrices $\mathbf{P}_1$ and $\mathbf{P}_2$, such that $\mathbf{P}_1\mathbf{C}\,\mathbf{P}_2$ is block-diagonal, with convolution matrices $\overline{\mathbf{C}}_m$ on the m^{th} diagonal block-entry [BD92]:

$$\mathbf{P}_1\mathbf{C}\,\mathbf{P}_2 \;=\; \begin{bmatrix} \ddots & & 0 \\ & \overline{\mathbf{C}}_m & \\ & & \overline{\mathbf{C}}_{m+1} \\ 0 & & \ddots \end{bmatrix} \triangleq \overline{\mathbf{C}} \tag{A.15}$$

Let $\mathbf{H}_{MF1}$ be the reconstruction matrix based on 1D matched filtering in both range and azimuth, then we have

$$\mathbf{H}_{MF1} \;=\; \mathbf{B}^*\mathbf{C}^* \;=\; \mathbf{B}^*(\mathbf{P}_1^t\mathbf{P}_1\mathbf{C}\,\mathbf{P}_2\mathbf{P}_2^t)^* \;=\; \mathbf{B}^*\mathbf{P}_2^t\overline{\mathbf{C}}^*\mathbf{P}_1^t \tag{A.16}$$

The matrix $\mathbf{B}^*$ represents the 1D discrete range compression operation and the matrix $\overline{\mathbf{C}}$ represents the 1D azimuth compression.

From section 2.4 we know that range migration compensation can be performed after the range compression. We show that this is also true if we use matrix algebra, not by deriving exact expressions, but by analyzing the matrix structures. Let $\Theta_{n-j} \triangleq \mathbf{B}_{n-j}\mathbf{B}_0^*$ and consider the following map

$$\mathbf{y} \;=\; \mathbf{u}\,\Theta_{j-n} \tag{A.17}$$

Let u_m be the m^{th} entry of $\mathbf{u}$, and let $\mathbf{e}_m$ be defined as

$$\mathbf{e}_m \;=\; [0 \cdots 0 \underset{\substack{\uparrow \\ m^{th}\ entry}}{1} 0 \cdots 0]$$

Then equation (A.17) can be expanded as

$$\mathbf{y} \;=\; \mathbf{u}\,\Theta_{j-n} \;=\; \mathbf{u}\,\mathbf{B}_{n-j}\mathbf{B}_0^* \;=\; \sum_m u_m \mathbf{e}_m \mathbf{B}_{n-j}\mathbf{B}_0^* \tag{A.18}$$

Observe that $\mathbf{e}_m\mathbf{B}_{n-j}$ yields the m^{th} row of $\mathbf{B}_{j-n}$. Define the finite vector

$$\mathbf{b}_{j-n;m} \;\triangleq\; [b_{0,j-n;m}\; b_{0,j-n;m}\; \cdots] \tag{A.19}$$

then

$$\mathbf{e}_m\mathbf{B}_{n-j} \;=\; [0 \;\cdots\; 0\; \mathbf{b}_{j-n;m}\; 0 \;\cdots\; 0] \tag{A.20}$$

From equations (A.3) and (A.7) we have

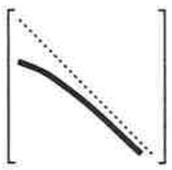
$$b_{i,j-n;m} \;=\; p(r_i - \Delta R(x_{j-n};\, R_m)) \tag{A.21}$$

Thus $\mathbf{b}_{i,j-n;m}$ is, in fact, the discrete shifted pulse, with shift $\Delta R(x_{j-n};\, R_m)$. Let $\boldsymbol{\theta}_{j-n;m} \triangleq \mathbf{e}_m\mathbf{B}_{n-j}\mathbf{B}_0^*$ and let $\mathcal{N}_{j-n;m}$ be the nearest integer to $|\Delta R(x_{j-n};\, R_m)|/dr$. Then $\boldsymbol{\theta}_{j-n;m}$ has peak value at the m^{th} position if $j - n = 0$, and at $(m - \mathcal{N}_{j-n;m})^{th}$ position for non-zero $j - n$. From equation (A.18) we have

$$\mathbf{e}_m\Theta_{j-n} \;=\; \boldsymbol{\theta}_{j-n;m} \tag{A.22}$$

Thus $\boldsymbol{\theta}_{j-n;m}$ is the m^{th} row of Θ_{j-n}

$$\Theta_{j-n} \;=\; \begin{bmatrix} \vdots \\ \boldsymbol{\theta}_{j-n;m} \\ \boldsymbol{\theta}_{j-n;m+1} \\ \vdots \end{bmatrix} \;= \tag{A.23}$$

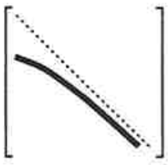


In equation (A.23) we have depicted the position of the peak by the drawn line, whereas the dotted line depicts the main diagonal. Observe that the peak position converges to the main diagonal for increasing m. Obviously, if m increases, R_m increases and thus the shift $|\Delta R(x_{j-n}; R_m)|$ decreases, see equation (2.28). At this point we can determine the matrix after range compression

$$\mathbf{S}\mathbf{B}^* = \begin{bmatrix} \ddots & \vdots & & 0 \\ \ddots & \mathbf{C}_1\mathbf{\Theta}_1 & \vdots & \\ & \mathbf{C}_0\mathbf{\Theta}_0 & \mathbf{C}_1\mathbf{\Theta}_1 & \\ & & \mathbf{C}_0\mathbf{\Theta}_0 & \ddots \\ 0 & & & \ddots \end{bmatrix} \tag{A.24}$$

The matrix $\mathbf{C}_{j-n}$ is a diagonal matrix, with $c_{j-n;m}$ on the $(m,m)^{th}$ entry. Thus

$$\mathbf{C}_{j-n}\mathbf{\Theta}_{j-n} = \begin{bmatrix} \vdots \\ c_{j-n;m}\,\theta_{j-n;m} \\ c_{j-n;m+1}\,\theta_{j-n;m+1} \\ \vdots \end{bmatrix} = \tag{A.25}$$



We are interested in the phase history of each point on the ground, which is in the components $c_{j-n;m}$. If we assume that the peak values of $\mathbf{\Theta}_{j-n}$ are sufficiently larger than non-peak values, then we may assume that the components $c_{j-n;m}$ now lie on the peak position of $\mathbf{C}_{j-n}\mathbf{\Theta}_{j-n}$. As we see in equation (A.25), in general the peak positions do not lie on the diagonal, except for the case $j - n = 0$.

Let $\widehat{\mathbf{C}}_{j-n}$ be $\mathbf{C}_{j-n}\mathbf{\Theta}_{j-n}$ for the $(m, m - \mathcal{N}_{j-n;m})^{th}$ entries (the peak values), and let the other entries of $\widehat{\mathbf{C}}_{j-n}$ be zero, and let $\widehat{\mathbf{C}}$ be defined as

$$\widehat{\mathbf{C}} \triangleq \begin{bmatrix} \ddots & \vdots & & 0 \\ \ddots & \widehat{\mathbf{C}}_1 & \vdots & \\ & \widehat{\mathbf{C}}_0 & \widehat{\mathbf{C}}_1 & \\ & & \widehat{\mathbf{C}}_0 & \ddots \\ 0 & & & \ddots \end{bmatrix} \tag{A.26}$$

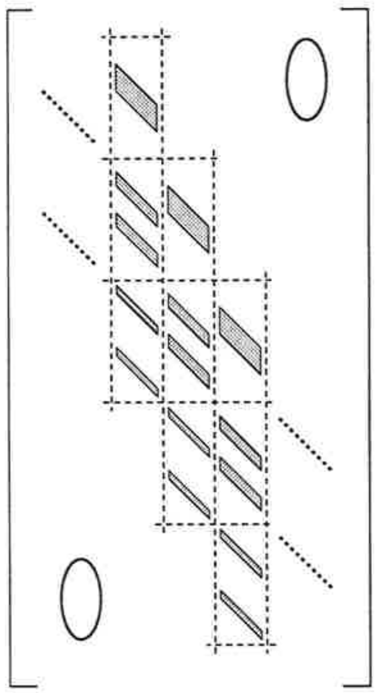


Figure A.1: The structure of $\overline{\overline{\mathbf{C}}}$. The block-entries are convolution matrices. The band structures are due to the range migration.

Then, using the permutation matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ as in equation (A.15) we obtain

$$\overline{\overline{\mathbf{C}}} \triangleq \mathbf{P}_1 \hat{\mathbf{C}} \mathbf{P}_2 = \begin{bmatrix} \ddots & & & 0 \\ \ddots & \overline{\overline{\mathbf{C}}}_{m,0} & & \\ & \overline{\overline{\mathbf{C}}}_{m,1} & \overline{\overline{\mathbf{C}}}_{m+1,0} & \\ & & \overline{\overline{\mathbf{C}}}_{m+1,1} & \ddots \\ 0 & & & \ddots \end{bmatrix} \tag{A.27}$$

The structure of this matrix is shown in figure A.1. Each "block-column" $\overline{\overline{\mathbf{C}}}_m$, defined as

$$\overline{\overline{\mathbf{C}}}_m \triangleq \begin{bmatrix} 0 \\ \overline{\overline{\mathbf{C}}}_{m,0} \\ \overline{\overline{\mathbf{C}}}_{m,1} \\ \vdots \\ 0 \end{bmatrix} \tag{A.28}$$

represents the convolution with the 2D impulse response of the SAR system after pulse compression, see equation (2.30). Let $\mathbf{H}_{MF2}$ be the reconstruction matrix that takes into account the range migration correction, then we have

$$\mathbf{H}_{MF2} = \mathbf{B}^*\widehat{\mathbf{C}}^* = \mathbf{B}^*(\mathbf{P}_1^t\mathbf{P}_1\widehat{\mathbf{C}}\mathbf{P}_2\mathbf{P}_2^t)^* = \mathbf{B}^*\mathbf{P}_2^t\overline{\overline{\mathbf{C}}}^*\mathbf{P}_1^t \tag{A.29}$$

APPENDIX B

DECONVOLUTION SYSTEMS

In chapter 2 we have taken the classical radar signal processing viewpoint to enhance resolution. A disadvantage of this viewpoint, however, is that it is somewhat outdated. For example, pulse compression, as we have described in section 2.3, is an inheritance from the days that radars only were used to *detect targets*. The main problem to solve was to detect as many as possible targets[1], such as aircraft, missiles and vehicles. Nowadays, we should formulate the problem in a more generic way, like a minimum mean square error problem: minimize the mean square error between the desired signal and the estimation obtained from an observed signal in the presence of noise. This might lead to a classical solution (for example a matched filter), but might also lead to novel solutions. In this appendix we consider the resolution problem as a deconvolution problem. We obtain a generic form of a deconvolution filter by means of modern linear algebra techniques, e.g., Singular Value Decomposition (SVD) and rank reduction [GvL89].

Linear Convolution Systems

We consider a linear system with input $[\cdots\ u_{-1}\ \boxed{u_0}\ u_1\ \cdots]$ and observed output signal $[\cdots\ y_{-1}\ \boxed{y_0}\ y_1\ \cdots]$. The box identifies the entry at the zeroth position. The relationship between the input and output signal is

$$y_n = \sum_{m=0}^{M-1} u_{n-m} a_m \tag{B.1}$$

[1]The term point *target* is also an example of this inheritance.

Thus the system performs a linear convolution operation of the signal $\mathbf{u}$ and a finite signal $\mathbf{a} = [a_0 \cdots a_{M-1}]$. Equation (B.1) can be written in matrix notation

$$\begin{bmatrix}\cdots & y_{-1} & \boxed{y_0} & y_1 & \cdots\end{bmatrix} = \begin{bmatrix}\cdots & u_{-1} & \boxed{u_0} & u_1 & \cdots\end{bmatrix}\begin{bmatrix}\ddots & & & & 0\\ & a_{M-1} & & & \\ \ddots & \vdots & a_{M-1} & & \\ & a_0 & \vdots & a_{M-1} & \\ & & \boxed{a_0} & \vdots & \ddots \\ & & & a_0 & \\ 0 & & & & \ddots\end{bmatrix} \tag{B.2}$$

The operator is banded and Toeplitz [GvL89].

In practical situations, however, linear convolution operation is limited: either the observed signal or the the input signal is windowed. Windowed observation signals occur typically in radar and acoustics, where the duration window is related to an observation interval. Windowed input signals typically occur in filter applications, in which finite signals are available. Consider both cases, thus assuming that either the observation window or the input window has length N, with $M \leq N$. Then, from equation (B.2), the following matrix equations are obtained

$$\begin{bmatrix}y_0 & \cdots & y_{N-1}\end{bmatrix} = \begin{bmatrix}u_{1-M} & \cdots & \boxed{u_0} & \cdots & u_{N-1}\end{bmatrix}\begin{bmatrix}a_{M-1} & & & 0\\ \vdots & \ddots & & \\ \boxed{a_0} & & \ddots & \\ & \ddots & & a_{M-1}\\ & & \ddots & \vdots \\ 0 & & & a_0\end{bmatrix} \tag{B.3}$$

$$\begin{bmatrix}y_0 & \cdots & y_{P-1}\end{bmatrix} = \begin{bmatrix}u_0 & \cdots & u_{N-1}\end{bmatrix}\begin{bmatrix}a_0 & \cdots & a_{M-1} & & 0\\ & \ddots & & \ddots & \\ & & \ddots & & \ddots \\ 0 & & a_0 & \cdots & a_{M-1}\end{bmatrix} \tag{B.4}$$

Equations (B.3) and (B.4) are *underdetermined* and *overdetermined*, respectively [GvL89].

The pseudo-inverse

The objective of this section is to find a deconvolution matrix that deconvolves either the under determined or overdetermined finite linear convolution system, if possible. Since we are mainly interested in radar echo signals we consider the underdetermined system. However, the techniques that we present in this section also apply to the overdetermined system. Let $\mathbf{u} = [u_{1-M} \cdots \boxed{u_0} \cdots u_{N-1}]$ and $\mathbf{y} = [y_0 \cdots y_{P-1}]$ and let $\mathbf{A}$ be the $N \times P$ underdetermined convolution matrix, then equation (B.3) becomes

$$\mathbf{y} \;=\; \mathbf{u}\,\mathbf{A} \tag{B.5}$$

Let $\mathbf{H}$ be an $P \times N$ matrix, such that $\hat{\mathbf{u}} = \mathbf{y}\,\mathbf{H}$ is an estimate of $\mathbf{u}$. We call $\mathbf{H}$ the deconvolution matrix. The optimum deconvolution matrix $\mathbf{H}$ is the one that minimizes the mean square error $\|\hat{\mathbf{u}} - \mathbf{u}\|^2$. The $\mathbf{H}$ that solves this minimization problem is the pseudo-inverse matrix $\mathbf{A}^+$ [GvL89]. In the overdetermined case the classical solution is often referred to as the *right-inverse* $\mathbf{A}^*(\mathbf{A}\,\mathbf{A}^*)^{-1}$. However, a requirement must then be that $\mathbf{A}$ has full rank. If $\mathbf{A}$ is singular then a natural substitute for the right-inverse is the pseudo-inverse. In [Fel93] the right-inverse is decomposed in a matched filter, represented by $\mathbf{A}^*$ and a side-lobe cancellation filter, represented by $(\mathbf{A}\,\mathbf{A}^*)^{-1}$.

In practical systems, however, $\mathbf{A}$ can be ill-conditioned, which makes the pseudo-inverse sensitive to noise. This can be shown as follows. Let σ_n, $n = 0, \cdots, N-1$ be the singular values of $\mathbf{A}$. Let $\omega_m = (2\pi/N)m$ and let $s_m = |A(j\omega_m)|$, $m = 0, \cdots, N-1$, be sample points of the power spectrum $A(j\omega)$ of $\mathbf{a}$. Then the singular values of $\mathbf{A}$ approach s_m if N approaches infinity, see [GS58] . Thus, if $N \to \infty$ then

$$\sigma_0 \;=\; \max\{s_0, \cdots, s_{N-1}\}, \qquad \sigma_{N-1} \;=\; \min\{s_0, \cdots, s_{N-1}\} \tag{B.6}$$

The condition $\kappa\{\mathbf{A}\}$ of $\mathbf{A}$ is defined as the ratio of its largest and smallest singular values

$$\kappa\{\mathbf{A}\} \;=\; \frac{\sigma_0}{\sigma_{N-1}} \;=\; \frac{\max\{s_0, \cdots, s_{N-1}\}}{\min\{s_0, \cdots, s_{N-1}\}} \tag{B.7}$$

If $\kappa\{\mathbf{A}\}$ is large ($\gg 1$) then the matrix $\mathbf{A}$ is called ill-conditioned. Most finite signals (except in case when $\mathbf{a}$ is white) have some stopband properties, so $\mathbf{A}$ is in most cases ill-conditioned.

The condition of a matrix is important in the presence of noise. Using an ill-conditioned matrix can decrease the signal-to-noise ratio dramatically. An SVD analysis will illustrate this. Let $\mathbf{A}$ has full column rank and let $\mathbf{A} = \mathbf{Q}\,\boldsymbol{\Sigma}\,\mathbf{V}^*$ be its SVD. Assume that $\mathbf{u}$ is a Wide Sense Stationary[2] (WSS) signal with zero mean and variance ν_u^2, and let $\mathbf{n}$ an additive noise signal with zero mean and variance ν_n^2. Let $\mathbf{u}$ and $\mathbf{n}$ be uncorrelated, and let the corrupted observation signal be $\hat{\mathbf{y}} = \mathbf{u}\,\mathbf{A} + \mathbf{n}$. The expected signal-to-noise ratio $\mathsf{SNR}_{\hat{\mathbf{y}}}$ is

$$\mathsf{SNR}_{\hat{\mathbf{y}}} \;=\; \frac{\mathsf{E}\|\mathbf{u}\,\mathbf{A}\|^2}{\mathsf{E}\|\mathbf{n}\|^2} \;=\; \frac{\mathsf{E}\|\mathbf{u}\,\mathbf{Q}\,\boldsymbol{\Sigma}\|^2}{N\nu_n^2} \;=\; \frac{1}{N}\frac{\nu_u^2}{\nu_n^2}\sum_{i=0}^{N-1}\sigma_i^2 \tag{B.8}$$

[2] A signal $\mathbf{u}$ is called Wide Sense Stationary signal if its expectation and second-moments are stationary, i.e., $\mathsf{E}u_n = constant$, for all n, and $\mathsf{E}u_n^* u_m = \mathsf{E}u_k u_{k+m-n}^*$, for all k, and it also satisfies $\mathsf{E}|u_n|^2 < \infty$, for all n.

where we used the fact that if $\mathbf{Q}$ is orthogonal and $\mathbf{u}$ is WSS with variance ν_u^2 and zero-mean, then $\mathbf{u}\,\mathbf{Q}$ is also WSS with variance ν_u^2 and zero-mean. Let $\hat{\mathbf{u}} = \hat{\mathbf{y}}\,\mathbf{A}^+ = \mathbf{u}\,\mathbf{A}\,\mathbf{A}^+ + \mathbf{n}\,\mathbf{A}^+$ be the deconvolved signal, then

$$\mathsf{SNR}_{\hat{\mathbf{u}}} = \frac{\mathsf{E}\|\mathbf{u}\,\mathbf{A}\,\mathbf{A}^+\|^2}{\mathsf{E}\|\mathbf{n}\,\mathbf{A}^+\|^2} = \frac{\mathsf{E}\|\mathbf{u}\,\mathbf{Q}\,\Sigma\,\Sigma^+\|^2}{\mathsf{E}\|\mathbf{n}\,\mathbf{V}\,\Sigma^+\|^2} = N\frac{\nu_u^2}{\nu_n^2}\left(\sum_{i=0}^{N-1}\sigma_i^{-2}\right)^{-1} \tag{B.9}$$

The increase of the signal-to-noise ratio after using the pseudo-inverse is then

$$\eta = \frac{\mathsf{SNR}_{\hat{\mathbf{u}}}}{\mathsf{SNR}_{\hat{\mathbf{y}}}} = N^2\left(\sum_{i=0}^{N-1}\sigma_i^2\right)^{-1}\left(\sum_{i=0}^{N-1}\sigma_i^{-2}\right)^{-1} \tag{B.10}$$

Observe that η solely depends on the singular values of $\mathbf{A}$. Obviously, if $\mathbf{A}$ is ill-conditioned, then the small singular values of $\mathbf{A}$ can cause a severe decrease of signal-to-noise ratio: $\eta \ll 1$.

Observe that we have found a measure for the signal-to-noise increase of a deconvolution operation. The deconvolution matrix, the pseudo-inverse $\mathbf{A}^+$, is in general not Toeplitz, corresponding to time variant filtering. If we interpret the set of singular values of $\mathbf{A}$ as a representation of the power spectrum[3] of the matrix $\mathbf{A}$, then the pseudo-inverse $\mathbf{A}^+$ can be interpreted as a matrix with an inverted power spectrum of $\mathbf{A}$.

Reduced rank estimation

A way to overcome this degradation η is to approximate $\mathbf{A}$ by a matrix with lower rank, see [GvL89]. Let $q < \text{rank}\{\mathbf{A}\}$, and let $\mathbf{A}_q$ the rank q matrix satisfying $\|\mathbf{A} - \mathbf{A}_q\| = \sigma_q$. Then $\mathbf{A}_q^+$ is an approximation of the pseudo-inverse $\mathbf{A}^+$. Let $\hat{\mathbf{u}}_q = \hat{\mathbf{y}}\,\mathbf{A}_q^+$, then

$$\mathsf{SNR}_{\hat{\mathbf{u}}_q} = \frac{\mathsf{E}\|\mathbf{u}\,\mathbf{A}\,\mathbf{A}_q^+\|^2}{\mathsf{E}\|\mathbf{n}\,\mathbf{A}_q^+\|^2} = q\frac{\nu_u^2}{\nu_n^2}\left(\sum_{i=0}^{q-1}\sigma_i^{-2}\right)^{-1} \tag{B.11}$$

We can now determine the signal-to-noise increase η_q if we perform the deconvolution with the reduced rank pseudo-inverse $\mathbf{A}_q^+$

$$\eta_q = \frac{\mathsf{SNR}_{\hat{\mathbf{u}}_q}}{\mathsf{SNR}_{\hat{\mathbf{y}}}} = qN\left(\sum_{i=0}^{q-1}\sigma_i^2\right)^{-1}\left(\sum_{i=0}^{N-1}\sigma_i^{-2}\right)^{-1} \tag{B.12}$$

Thus $\eta_q \leq \eta$.

[3]The spectrum of a square matrix is defined as the set of eigenvalues of the matrix [GvL89]. If we consider the autocorrelation matrix $\mathbf{R} = \mathbf{A}^*\mathbf{A}$, then the set of eigenvalues of $\mathbf{R}$ coincides with the set of squared singular values of $\mathbf{A}$.

In [Sch91] a method is described for finding a trade-off between mean square signal error and noise sensitivity. Let $\hat{\mathbf{u}}_q = \hat{\mathbf{y}}\,\mathbf{A}_q^+$ and consider the mean square error

$$\mathsf{E}\|\hat{\mathbf{u}}_q - \mathbf{u}\|^2 \;=\; \mathsf{E}\|\mathbf{u}(\mathbf{A}\,\mathbf{A}_q^+ - \mathbf{I})\|^2 + \mathsf{E}\|\mathbf{n}\,\mathbf{A}_q^+\|^2 \;\triangleq\; \epsilon_s(q) + \epsilon_n(q) \tag{B.13}$$

The terms $\epsilon_s(q)$ and $\epsilon_n(q)$ can be written in terms of singular values σ_i and variances ν_u^2 and ν_n^2 as follows

$$\begin{aligned} \epsilon_s(q) &= \mathsf{E}\|\mathbf{u}\,\mathbf{Q}(\boldsymbol{\Sigma}\,\boldsymbol{\Sigma}_q^+ - \mathbf{I})\|^2 \;=\; (P-q)\nu_u^2 & \text{(B.14)}\\ \epsilon_n(q) &= \mathsf{E}\|\mathbf{n}\,\mathbf{V}\,\boldsymbol{\Sigma}_q^+\mathbf{Q}\|^2 \;=\; \nu_n^2\sum_{i=0}^{q-1}\sigma_i^{-2} & \text{(B.15)} \end{aligned}$$

Then the optimum q minimizes the mean square error

$$\min_q \epsilon_s(q) + \epsilon_n(q) \tag{B.16}$$

The solution $\hat{\mathbf{u}}_q$ is referred to as the reduced rank estimate.

Time-invariant deconvolution filters

A problem not addressed yet is the dimension of the system. In practice M and N can be $\mathcal{O}(10^3)$ and $\mathcal{O}(10^4)$, respectively. This means that the computation of $\mathbf{A}_q^+$ is hardly possible. Moreover, since we deal with pulsed radar signals, several echo signals $\hat{\mathbf{y}}$ must be processed subsequently. Time-invariant filtering would then be preferable. This is possible, as we shall now show. Given the $P \times N$ convolution matrix $\mathbf{A}$

$$\mathbf{A} \;=\; \begin{bmatrix} a_{M-1} & & & 0 \\ \vdots & \ddots & & \\ \boxed{a_0} & & \ddots & \\ & \ddots & & a_{M-1} \\ & & \ddots & \vdots \\ 0 & & & a_0 \end{bmatrix} \tag{B.17}$$

Let N' be an integer that satisfies $M \le N' \le N$ and let $P' = N' + M - 1$. We can define then a $P' \times N'$ convolution matrix $\underline{\mathbf{A}}$, that slides along the diagonal of $\mathbf{A}$

$$\mathbf{A} \;=\; \begin{bmatrix} \ddots & & 0 \\ & \boxed{\underline{\mathbf{A}}} & \\ 0 & & \ddots \end{bmatrix} \tag{B.18}$$

Then, using the Hankel matrices

$$\hat{\mathbf{Y}} = \begin{bmatrix} \hat{y}_0 & \hat{y}_1 & \cdots & \hat{y}_{N'-1} \\ \hat{y}_1 & \hat{y}_2 & \cdots & \hat{y}_{N'} \\ \vdots & \vdots & & \vdots \\ \hat{y}_{N-N'} & \hat{y}_{N-N'+1} & \cdots & \hat{y}_{N-1} \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} u_{1-M} & \cdots & \boxed{u_0} & u_1 & \cdots & u_{N'-1} \\ u_{2-M} & \cdots & u_1 & u_2 & \cdots & u_{N'} \\ \vdots & & \vdots & \vdots & & \vdots \\ u_{N-P'} & \cdots & u_{N-N'} & u_{N-N'+1} & \cdots & u_{N-1} \end{bmatrix}$$

$$\mathbf{N} = \begin{bmatrix} n_0 & n_1 & \cdots & n_{N'-1} \\ n_1 & n_2 & \cdots & n_{N'} \\ \vdots & \vdots & & \vdots \\ n_{N-N'} & n_{N-N'+1} & \cdots & n_{N-1} \end{bmatrix}$$

we can rewrite the matrix notation of the convolution problem of equation (B.3), including the additive noise, as

$$\hat{\mathbf{Y}} = \mathbf{U}\underline{\mathbf{A}} + \mathbf{N} \tag{B.19}$$

Instead of a linear system with input and output vectors, we have constructed an equivalent linear system with input and output matrices. The dimension of the convolution matrix $\underline{\mathbf{A}}$, however, is much smaller than the dimension of $\mathbf{A}$.

The reduced rank deconvolution matrix $\underline{\mathbf{A}}_q^+$ is now obtained by minimizing the mean square error $\mathsf{E}\|\hat{\mathbf{U}}_q - \mathbf{U}\|_F^2 \triangleq \epsilon_s(q) + \epsilon_n(q)$ over all $0 \leq q < \text{rank}\{\underline{\mathbf{A}}\}$. The reduced rank estimation is now a matrix rather than a row vector

$$\hat{\mathbf{U}}_q = \begin{bmatrix} \hat{u}_{1-M}^{(1-M)} & \cdots & \boxed{\hat{u}_0^{(0)}} & \hat{u}_1^{(1)} & \cdots & \hat{u}_{N'-1}^{(N'-1)} \\ \hat{u}_{2-M}^{(1-M)} & \cdots & \hat{u}_1^{(0)} & \hat{u}_2^{(1)} & \cdots & \hat{u}_{N'}^{(N'-1)} \\ \vdots & & \vdots & \vdots & & \vdots \\ \hat{u}_{N-P'}^{(1-M)} & \cdots & \hat{u}_{N-N'}^{(0)} & \hat{u}_{N-N'+1}^{(1)} & \cdots & \hat{u}_{N-1}^{(N'-1)} \end{bmatrix} \tag{B.20}$$

Observe that $\hat{\mathbf{U}}$ does not have Hankel structure, whereas $\mathbf{U}$ has Hankel structure. Structure preservation may be an additional constraint to the estimation problem. It is beyond our scope to investigate the implications of this constraint to the estimation problem. We refer to [CHV95] for an approach that solves a similar problem with preservation of the structure.

If we assume that for all samples u_k, the estimate $\hat{u}_k^{(p)}$ is sufficiently accurate for all p, then we might as well take the p^{th} column of $\hat{\mathbf{U}}_q$ as an estimate of the p^{th} column of $\mathbf{U}$. Let $\hat{\underline{\mathbf{u}}}_q^{(p)}$ and

$\underline{\mathbf{a}}_q^{(p)}$ denote the p^{th} rows of $\widehat{\mathbf{U}}_q^t$ and $\underline{\mathbf{A}}_q^t$, respectively. Then

$$\widehat{\underline{\mathbf{u}}}_q^{(p)} = \underline{\mathbf{a}}_q^{(p)} \widehat{\mathbf{Y}}^t \tag{B.21}$$

Let $\underline{a}_q^{(p,k)} \triangleq [\underline{\mathbf{a}}_q^{(p)}]_k$, $k = 0, \cdots, N' - 1$. Equation (B.21) represents a in fact a convolution operation. Then, after rearranging, it can be written as

$$[\hat{u}_l^{(p)} \cdots \hat{u}_{l+N-N'}^{(p)}] = [\hat{y}_0 \cdots \hat{y}_{N-1}] \begin{bmatrix} \underline{a}_q^{(p,0)} & & 0 \\ \vdots & \ddots & \\ \underline{a}_q^{(p,N'-1)} & \ddots & \\ & \ddots & \underline{a}_q^{(p,0)} \\ & \ddots & \vdots \\ 0 & & \underline{a}_q^{(p,N'-1)} \end{bmatrix} \tag{B.22}$$

Here we recognize a convolution matrix, and thus we may write equation (B.22) as a convolution operation

$$\widehat{\underline{\mathbf{u}}}_q^{(p)} = \underline{\mathbf{a}}_q^{(p)\leftrightarrow} * \widehat{\mathbf{y}} \tag{B.23}$$

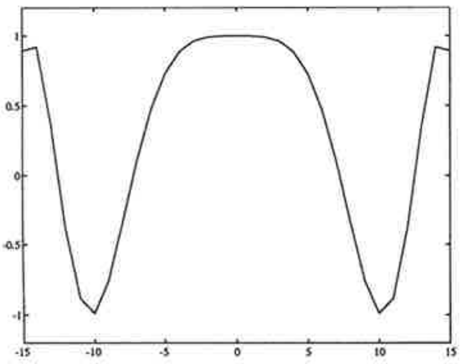
where $\underline{\mathbf{a}}_q^{(p)\leftrightarrow}$ is $\underline{\mathbf{a}}_q^{(p)}$ with entries in reversed order. Thus we have derived a time-invariant deconvolution filter of length N'. The solution $\widehat{\underline{\mathbf{u}}}_q^{(p)}$ is as optimum as the solution obtained from the deconvolution matrix $\underline{\mathbf{A}}_q^+$ in terms of mean square error. The application of a deconvolution-filter, however, is more efficient in terms of computational complexity compared to the deconvolution matrix.

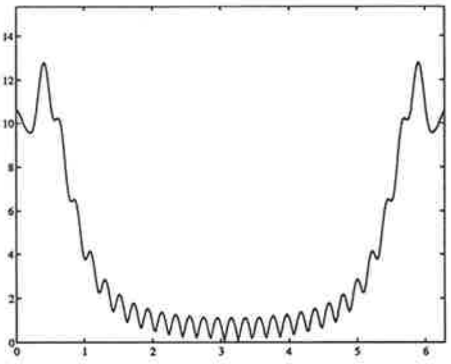
A worked example

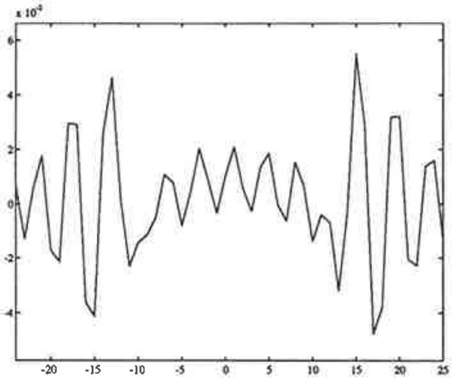
We terminate this section with an example to illustrate the techniques that we have developed above. Consider the discrete chirp signal $\mathbf{a}$ of length $M = 31$, see figure B.1.a, and its spectrum in figure B.1.b. Construct an $P \times N$ convolution matrix $\mathbf{A}$ where we choose $N = 100$ (and thus $P = 130$). The observed signal $\widehat{\mathbf{y}} = \mathbf{u}\,\mathbf{A} + \mathbf{n}$ has length 100. For the analysis of the signal-to-noise ratio we assume that both input signal $\mathbf{u}$ and noise $\mathbf{n}$ are WSS with variance $\nu_n^2 = 2\nu_u^2$. Using the pseudo-inverse matrix $\mathbf{A}^+$ to estimate the input signal results in a severe decrement of the expected signal-to-noise ratio: $\eta = 0.036$. Thus we reduce the rank of the pseudo-inverse, which leads to an optimized reduced rank pseudo-inverse $\mathbf{A}_q^+$, where the optimized reduced rank is $q = 56$. The decrement of the expected signal-to-noise ratio is now $\eta_q = 0.52$, which is approx. a factor 14 less than η.

Suppose that, for some reason, the dimension of $\mathbf{A}$ is too large. Then we can construct an $N' \times P'$ convolution matrix $\underline{\mathbf{A}}$, where we choose $N' = 40$ (and thus $P' = 70$). Again we reduce the rank of the pseudo-inverse, leading to $\underline{\mathbf{A}}_q^+$, where, in this case, $q = 23$. The reduced rank time-invariant deconvolution filter is then $\underline{\mathbf{a}}_q^{(p)\leftrightarrow}$, where, in this example, $p = 35$. The filter $\underline{\mathbf{a}}_q^{(p)\leftrightarrow}$ and its spectrum are shown in figure B.1.c and B.1.d. Observe that the spectrum of $\underline{\mathbf{a}}_q^{(p)\leftrightarrow}$ estimates the inverse spectrum of $\mathbf{a}$ within a certain band. The bandwidth is approximately determined by the q largest singular values of $\underline{\mathbf{A}}$, using the relationship between the SVD and the spectrum [GS58].

The results of the estimators are shown in figure B.2. As input signal we have used two dirac signals. The dotted lines are the estimates in absence of additive noise. The straight lines are the results in the presence of additive noise. For completeness we also add the estimates achieved with the matched filter. Observe that this estimate is superior in the terms of signal-to-noise increment, but the two dirac signals cannot be distinguished. The signal-to-noise decrement as a result of the pseudo-inverse is clearly seen. The reduced rank estimates are both compromises between the estimates obtained by using the matched filter and the pseudo-inverse.


(a)


(b)



(c)

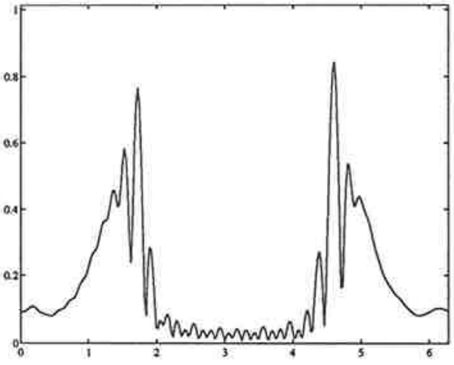

(d)

Figure B.1: The real part of the discrete chirp signal **a** (a), its spectrum (b), the reduced rank time-invariant filter $\underline{\mathbf{a}}_q^{(p)}$ (c) and its spectrum (d).

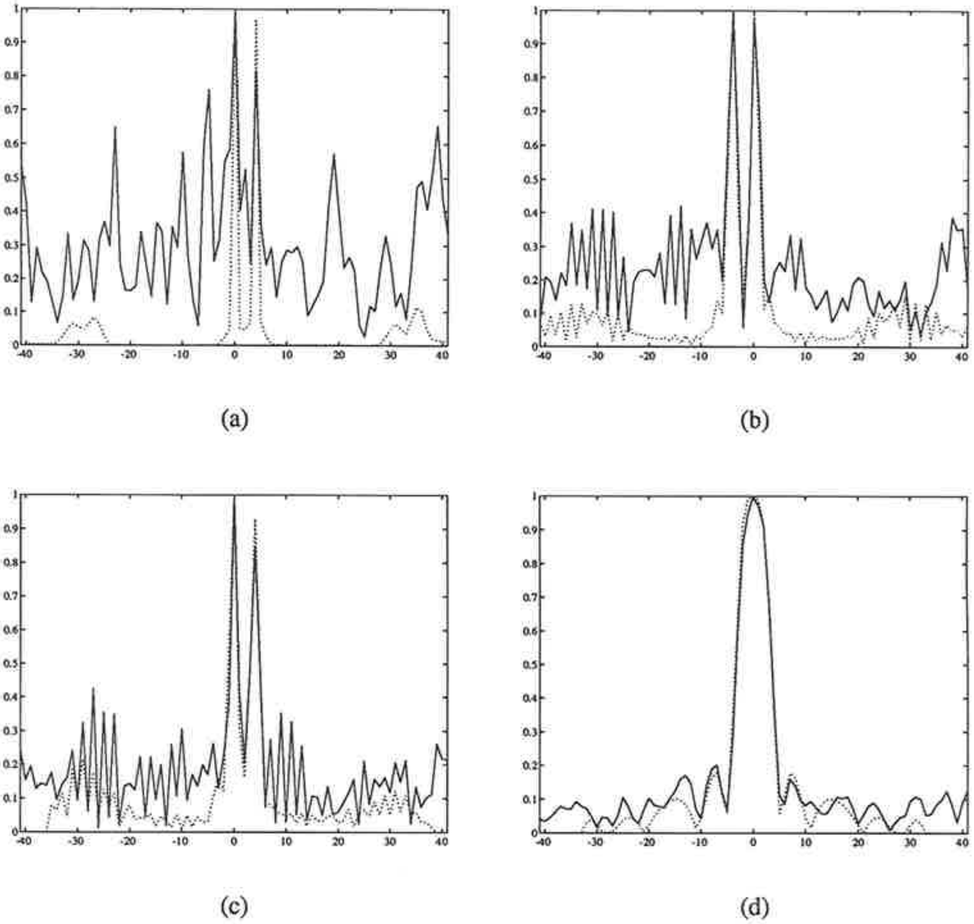


Figure B.2: Four estimates of two input dirac signals in the absence (dotted lines) and presence (straight line) of additive noise, obtained by the pseudo-inverse (a), the reduced rank pseudo-inverse (b), the reduced-rank shift invariant deconvolution filter (c) and the matched filter (d).

APPENDIX C

MOTION COMPENSATION

In chapters 2 and 3 we have assumed that the flight trajectory was ideal, in that v is constant and the platform moves along a straight line. In general, this will not be the case, especially for airborne SAR. For example, due to air turbulence the flight trajectory will be some high order polynomial around the ideal straight line. To model the corresponding phase error we start with an analysis of the distorted range $\hat{R}(t)$ [Oli89]. In this section we derive the all formulations for the time continuous case. The extension to the discrete case is straight-forward. Without lack of generality we set $\gamma = 0$. We assume that the error $\hat{R}(t) - R(t)$ can be expressed in an along-track error $\hat{x}(t) - x(t)$ along the x-axis and an across-track error $r(t)$ in the yz-plane, see figure C.1. Furthermore, the nominal range corresponding to the distorted flight trajectory is $\hat{R}_0 = R_0 + \Delta R_0$. From figure C.1 we see that

$$\hat{R}(t) \;=\; \sqrt{(\hat{R}_0 + r(t))^2 + \hat{x}(t)^2} \tag{C.1}$$

If $|\hat{x}(t)| \ll \hat{R}_0 + r(t)$ then equation (C.2) approximates to

$$\hat{R}(t) \;=\; \hat{R}_0 + r(t) + \frac{\hat{x}(t)^2}{2\hat{R}_0} \;\simeq\; r(t) + \frac{\hat{x}(t)^2}{2\hat{R}_0} \tag{C.2}$$

where $\Delta R_0 = \hat{R}_0 - R_0$ is the error in the nominal range.

Across-track motion compensation

Within the SAR processing chain we can divide the motion compensation in two parts: across-track motion compensation and along-track motion compensation. At this point we assume

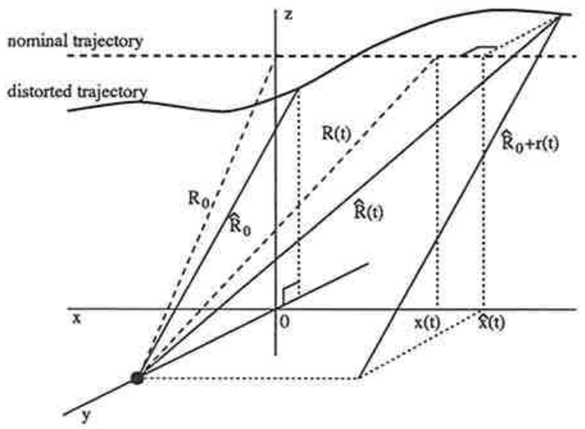


Figure C.1: Ideal (dashed) and distorted (straight) SAR geometry. It is assumed that the antenna is at position $x(t)$ and has distance $R(t)$ to a specific point target with nominal range R_0. The actual situations is that the antenna position is $\hat{x}(t)$, has distance $\widehat{R}(t)$ to the point target, which has a nominal range $\widehat{R}_0$.

that the distorted flight trajectory is known. Recall the 2D echo signal after pulse compression, equation (2.30), and include the motion parameters, assuming that range migration is negligible[1]

$$\hat{g}(t',t) = \operatorname{sinc}(\alpha\tau_p(t'-2r(t)/c))\exp(j\hat{\varphi}(t)) \tag{C.3}$$

The distorted phase is

$$\hat{\varphi}(t) = -\frac{4\pi}{\lambda}\left(r(t)+\frac{\hat{x}(t)^2}{2\widehat{R}_0}\right) \tag{C.4}$$

Let $\hat{\varphi}_r(t)$ and $\hat{\varphi}_a(t)$ be the phase terms due to the across-track motion and the along-track motion, respectively, such that $\hat{\varphi}(t) = \hat{\varphi}_r(t) + \hat{\varphi}_a(t)$, then

$$\hat{\varphi}_r(t) = -\frac{4\pi}{\lambda}r(t), \qquad \hat{\varphi}_a(t) = -\frac{4\pi}{\lambda}\frac{\hat{x}(t)^2}{2\widehat{R}_0}$$

Observe that the peak of the sinc-function varies with the distorted trajectory in range $r(t)$, see figure C.2. Thus we must delay $\hat{g}(t',t)$ in t with $2r(t)/c$ seconds. In practice, this means that the discrete signal for each azimuth instant j must be interpolated in range. Follows the phase correction in range using the across-track phase error $\varphi_r(t)$. Let $\hat{g}'(t',t)$ be the echo signal after the across-track motion compensation

$$\hat{g}'(t',t) = \hat{g}(t'+2r(t)/c,t)\exp(-j\hat{\varphi}_r(t)) = \operatorname{sinc}(\alpha\tau_p t')\exp(j\hat{\varphi}_a(t)) \tag{C.5}$$

[1] We have omitted the constant $\exp(-j4\pi/c\widehat{R}_0)$.

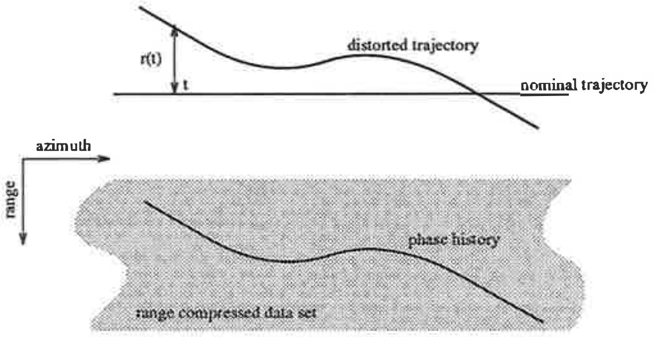


Figure C.2: The variation of the echo peak with the distorted trajectory in range $r(t)$ after pulse compression.

Along-track motion compensation

Remains the along-track motion compensation of $\hat{g}'(t', t)$. Let $\Delta x(t) = \hat{x}(t) - x(t)$, then it is desirable to express the distorted along-track phase

$$\hat{\varphi}_a(t) = -\frac{4\pi}{\lambda}\frac{\hat{x}(t)^2}{2\widehat{R}_0} = -\frac{4\pi}{\lambda}\frac{(x(t)+\Delta x(t))^2}{2R_0} \tag{C.6}$$

This term can be obtained by approximation of the term $\hat{x}(t)^2/\widehat{R}_0$, as we will show now. Observe that

$$\widehat{R}_0^{-1} = R_0^{-1}\left(\sqrt{1+\frac{\Delta R_0}{R_0}}\right)^2 \tag{C.7}$$

If $|\Delta R_0| \ll R_0$ then equation (C.7) approximates to

$$\widehat{R}_0^{-1} = R_0^{-1}\left(1-\frac{\Delta R_0}{2R_0}\right)^2 \tag{C.8}$$

We can also assume that the distortion $\Delta x(t) = \hat{x}(t) - x(t)$ is caused by a distortion in the nominal speed: $\hat{v}(t) - v$. Define

$$\Delta v \triangleq \hat{v}(0) - v, \quad \dot{v} \triangleq \frac{\mathrm{d}}{\mathrm{d}t}\hat{v}(t)\Big|_{t=0}, \quad \ddot{v} \triangleq \frac{\mathrm{d}^2}{\mathrm{d}t^2}\hat{v}(t)\Bigg|_{t=0}$$

and recall that $x(t) = vt$, then we can write

$$\begin{aligned}\hat{x}(t) = \hat{v}t &= (v+\Delta v)t + \tfrac{1}{2}\dot{v}t^2 + \tfrac{1}{6}\ddot{v}t^3 + \cdots \\ &= x(t)\left(1+\frac{\Delta v}{v}+\frac{\dot{v}}{2v}t+\frac{\ddot{v}}{6v}t^2+\cdots\right)\end{aligned} \tag{C.9}$$

Assume that $\Delta v \Delta R_0 \ll vR_0$, $\dot{v}\Delta R_0 \ll vR_0$ and $\ddot{v}\Delta R_0 \ll vR_0$, then, from equations (C.8) and (C.9), we have that $\hat{x}(t)^2/\widehat{R}_0$ approximates to

$$\frac{\hat{x}(t)^2}{\widehat{R}_0} = \frac{\hat{x}(t)^2}{R_0}\left(1-\frac{\Delta R_0}{2R_0}\right)^2 = \frac{x(t)^2}{R_0}\left(1-\frac{\Delta R_0}{2R_0}+\frac{\Delta v}{v}+\frac{\dot{v}}{2v}t+\frac{\ddot{v}}{6v}t^2+\cdots\right)^2 \qquad \text{(C.10)}$$

Or,

$$\Delta x(t) = x(t)\left(\frac{\Delta v}{v}-\frac{\Delta R_0}{2R_0}+\frac{\dot{v}}{2v}t+\frac{\ddot{v}}{6v}t^2+\cdots\right) \qquad \text{(C.11)}$$

Thus, indeed the distorted along-track phase approximates to

$$\hat{\varphi}_a(t) = -\frac{4\pi}{\lambda}\frac{(x(t)+\Delta x(t))^2}{2R_0} = -\frac{4\pi}{\lambda}\frac{v^2(t+\Delta x(t)/v)^2}{2R_0} \qquad \text{(C.12)}$$

We know that the ideal phase of the echo signal would be

$$\varphi_a(t) = -\frac{4\pi}{\lambda}\frac{x(t)^2}{2R_0} = \hat{\varphi}_a(t-\Delta x(t)/v) \qquad \text{(C.13)}$$

Thus $\hat{g}'(t',t)$ can be expressed as

$$\hat{g}'(t',t) = \operatorname{sinc}(\alpha\tau_p t')\exp(j\varphi_a(t+\Delta x(t)/v)) = g(t',t+\Delta x(t)/v) \qquad \text{(C.14)}$$

In practice, this means that the discrete signal must be interpolated in azimuth using the map $t_j \leftarrow t_j - \Delta x(t_j)/v$.

A requirement for motion compensation is that $r(t)$ and $\Delta x(t)$ are known, either by measurement (using motion and position sensors like INS and GPS) or by estimation. The along-track position error $\Delta x(t)$ depends on ΔR_0, Δv and the first, second, etc., derivates of $\hat{v}(t)$. In general, ΔR_0 can be eliminated by the exact measurement of the time of pulse transmission and start time of the A/D conversion of the received echo. The along-track speed error components can be obtained by an exact measurement of the x-position locked on a time reference. The across-track position error can be obtained from the across-track position measurement.

Autofocusing

Estimation of the motion parameters is commonly referred to as autofocusing. Above we have determined the across-track motion error $r(t)$ and the along-track motion error $\Delta x(t)$. If these errors are not known, then still the motion compensation can be performed. In that case we approximate the distorted phase $\hat{\varphi}(t)$ by a series in t. Its coefficients can then be estimated from the echo data.

The first step is to derive expressions for these coefficients. Observe that we can write $r(t) = v_r(t)t$, where $v_r(t)$ is referred to as the across-track speed distortion. Expand $v_r(t)$ around $t = 0$, as we did before with $\hat{v}(t)$, then we find

$$r(t) \;=\; v_r t + \tfrac{1}{2}\dot{v}_r t^2 + \tfrac{1}{6}\ddot{v}_r t^3 + \cdots \tag{C.15}$$

Then, substitution of equations (C.15) and (C.10) in the distorted phase, see equation (C.4), and collecting the third-order and higher-order components in the series $\zeta(t)$, gives

$$\hat{\varphi}(t) \;=\; -\frac{4\pi}{\lambda}\left(v_r t + \frac{v^2}{2R_0}\left(1 - \frac{\Delta R_0}{R_0} + \frac{2\Delta v}{v} + \frac{\dot{v}_r R_0}{v^2}\right)t^2 + \zeta(t)\right) \tag{C.16}$$

Two remarks are in order here:

- The across-track speed distortion introduces a Doppler centroid

$$f_{dc} \;=\; \frac{1}{2\pi}\frac{\mathrm{d}}{\mathrm{d}t}\hat{\varphi}(t)\bigg|_{t=0} \;=\; -\frac{2v_r}{\lambda} \tag{C.17}$$

  In fact, this Doppler centroid may also contain a component as a result of an unknown squint angle.

- The quadratic term in $\hat{\varphi}(t)$ approximates to

$$\frac{v^2}{2R_0}\left(1 - \frac{\Delta R_0}{R_0} + \frac{2\Delta v}{v} + \frac{\dot{v}_r R_0}{v^2}\right) \;=\; \frac{(v + \Delta v')^2}{2R_0} \tag{C.18}$$

  if the following condition holds

$$\left|-\frac{\Delta R_0}{R_0} + \frac{2\Delta v}{v} + \frac{\dot{v}_r R_0}{v^2}\right| \;\le\; \frac{|\Delta R_0|}{R_0} + \frac{2|\Delta v|}{v} + \frac{|\dot{v}_r| R_0}{v^2} \;\ll\; 1 \tag{C.19}$$

  In that case we have

$$\Delta v' \;=\; v\sqrt{1 - \frac{\Delta R_0}{R_0} + \frac{2\Delta v}{v} + \frac{\dot{v}_r R_0}{v^2}} - v \;=\; \Delta v - \frac{v\Delta R_0}{2R_0} + \frac{\dot{v}_r R_0}{2v} \tag{C.20}$$

With equations (C.17) and (C.20), we can write equation (C.16) as

$$\hat{\varphi}(t) \;=\; -\frac{4\pi}{\lambda}\left(-\tfrac{1}{2}\lambda f_{dc} t + \frac{(v + \Delta v')^2 t^2}{2R_0} + \zeta(t)\right) \tag{C.21}$$

The unknown parameters in the zeroth and the first order coefficients are thus f_{dc} and $\Delta v'$, and are also called the *autofocus parameters*. It is not necessary to obtain more autofocus parameters for the estimation the higher order coefficients, as we show by the *mapdrift autofocus method* that we discuss hereafter.

Mapdrift autofocus

Most autofocusing techniques are based on mapdrift, which we now describe briefly. Mapdrift autofocusing is sufficient for slow varying motion errors. Other autofocusing approaches, such as contrast optimization, speckle processing, Wigner-Ville distribution method, binary multiplexing and reflectivity displacement method are proposed in [FW85, BG88, EGCJ89, Mor90]. Since it goes beyond the scope of this thesis to describe them we refer to these references for details and evaluations. An evaluation and comparison of some methods are given in e.g. [GB88, BBOW92, Ott91].

Let the distorted azimuth echo signal be given as

$$\hat{g}(t) \;=\; \exp(j\hat{\varphi}(t)) \;=\; \exp\left(-j\frac{4\pi}{\lambda}\left(-\tfrac{1}{2}\lambda f_{dc}t + \frac{(v+\Delta v')^2 t^2}{2R_0} + \zeta(t)\right)\right) \qquad \text{(C.22)}$$

where we have omitted t and R_0 for convenience. Then the uncompensated azimuth compressed signal $\hat{r}(t)$ obeys

$$\begin{aligned}
\hat{r}(t) &= \int_{-\frac{1}{2}T_a}^{\frac{1}{2}T_a} \hat{g}(\tau)a(t-\tau)\mathrm{d}\tau \\
&= \int_{-\frac{1}{2}T_a}^{0} \hat{g}(\tau)a(t-\tau)\mathrm{d}\tau + \int_{0}^{\frac{1}{2}T_a} \hat{g}(\tau)a(t-\tau)\mathrm{d}\tau \\
&= \hat{r}_1(t) + \hat{r}_2(t) \qquad \text{(C.23)}
\end{aligned}$$

The mapdrift approach is based on the fact that the peaks of $\hat{r}_1(t)$ and $\hat{r}_2(t)$ are mutually displaced, say with a factor ΔT, which is proportional to $\Delta v'$ [BG88, Ott89]. This can be shown by expanding $\hat{r}_1(t)$ as follows

$$\begin{aligned}
\hat{r}_1(t) &= \int_{-\frac{1}{2}T_a}^{0} \exp\left(-j\frac{4\pi}{\lambda}\left(-\tfrac{1}{2}\lambda f_{dc}\tau + \frac{(v+\Delta v')^2\tau^2}{2R_0} + \zeta(\tau)\right)\right) \exp\left(j\frac{4\pi}{\lambda}\frac{v^2(t-\tau)^2}{2R_0}\right)\mathrm{d}\tau \\
&\simeq \int_{-\frac{1}{4}T_a}^{\frac{1}{4}T_a} \exp\left(-j\frac{4\pi}{\lambda}\left(\frac{2v\Delta v' + \Delta v'^2}{2R_0}\tau^2 + \zeta(\tau - \tfrac{1}{4}T_a)\right)\right) \\
&\times\; \exp\left(-j\frac{4\pi}{\lambda}\frac{2tv^2 - \frac{1}{2}(2v\Delta v' + \Delta v'^2)T_a - \lambda f_{dc}R_0}{2R_0}\tau\right)\mathrm{d}\tau \qquad \text{(C.24)}
\end{aligned}$$

The integral in equation (C.24) cannot be computed analytically, but it is clear that $|\hat{r}_1(t)|$ is maximum if

$$2tv^2 - \tfrac{1}{2}(2v\Delta v' + \Delta v'^2)T_a - \lambda f_{dc}R_0 \;=\; 0 \qquad \text{(C.25)}$$

which occurs at

$$t \;=\; t_1 \;=\; \frac{2v\Delta v' + \Delta v'^2}{2v^2}\tfrac{1}{2}T_a + \frac{\lambda f_{dc}R_0}{2v^2} \qquad \text{(C.26)}$$

Doing the same analysis on $\hat{r}_2(t)$, we find that $|\hat{r}_2(t)|$ is maximum at

$$t = t_2 = -\frac{2v\Delta v' + \Delta v'^2}{2v^2}\tfrac{1}{2}T_a + \frac{\lambda f_{dc} R_0}{2v^2} \tag{C.27}$$

It thus follows that

$$\Delta T = t_1 - t_2 = \frac{2v\Delta v' + \Delta v'^2}{v^2}\tfrac{1}{2}T_a \tag{C.28}$$

Once ΔT is obtained from evaluating $\hat{r}_1(t)$ and $\hat{r}_2(t)$, $\Delta v'$ is readily obtained as

$$\Delta v' = -v\left(1 \pm \sqrt{1 + \frac{\Delta T}{\frac{1}{2}T_a}}\right) \tag{C.29}$$

If $\Delta T \ll \frac{1}{2}T_a$, then $\Delta v'$ approximates to[2]

$$\Delta v' \approx \frac{\Delta T}{T_a}v \tag{C.30}$$

As we have shown, the mapdrift based approach is based on the fact that motion parameters are determined by an error $\Delta v'$ in the nominal speed. By subdividing the aperture $(-\frac{1}{2}T_a, \frac{1}{2}T_a)$ into two so-called *subapertures* $(-\frac{1}{2}T_a, 0)$ and $(0, \frac{1}{2}T_a)$, we can obtain two SAR images (so-called *looks*), which have a mutual displacement ΔT, which can be measured. Via some algebraic manipulations it was shown that ΔT is proportional to the error $\Delta v'$. In fact, if we divide the aperture into more subapertures, higher order errors in the nominal speed can be determined. Generally, the order of the estimated error is one less than the number of subapertures used. An extensive discussion of a mapdrift based method with arbitrary number of subapertures is given in [BG88]. However, increasing the number of subapertures implies a decrement of bandwidth (and thus resolution) per look, leading to a more and more inaccurate estimate of ΔT.

It remains to estimate the Doppler centroid offset f_{dc}. Observing that

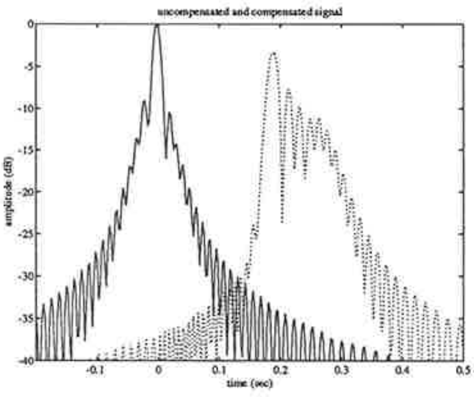
$$t_1 + t_2 = \frac{\lambda f_{dc} R_0}{v^2} \tag{C.31}$$

the Doppler centroid obeys

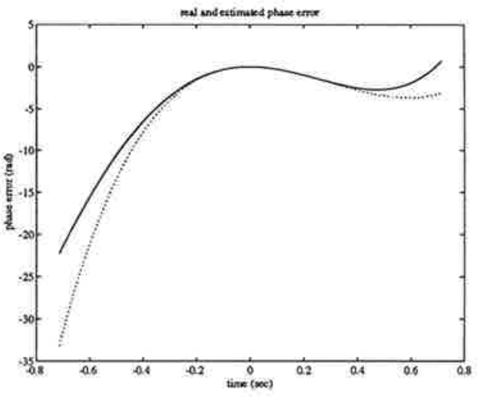
$$f_{dc} = (t_1 + t_2)\frac{v^2}{\lambda R_0} \tag{C.32}$$

More details about specific Doppler centroid estimation methods and their evaluations are given in [LHCW85, Jin86, Mad89]. It should be stated that Doppler centroid estimation as an isolated problem only occurs in satellite SAR processing, since the satellite orbit is relatively stable in

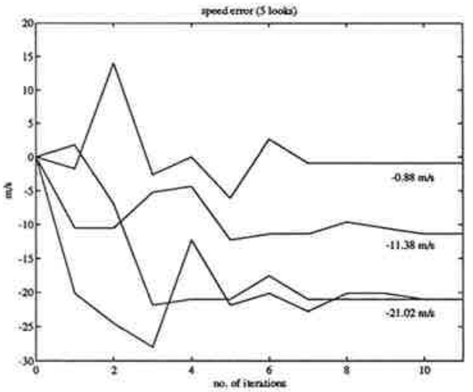
[2]We assume that the solution $\Delta v = -2v - v\Delta T/T_a$ makes no sense.



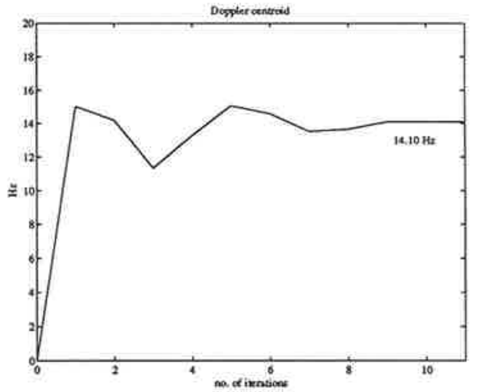
(a)



(b)



(c)



(d)

Figure C.3: Simulated compensated (straight line) and uncompensated (dotted line) impulse responses (a), the real phase error (straight line) and the phase error (dotted line) estimated using mapdrift autofocusing with 5 looks (b), the estimated speed errors between successive looks (c) and the estimated Doppler centroid (d).

terms of across-track and along-track motions. In airborne SAR the main problem will almost always be the actual motion compensation as we have discussed in this section.

A simulated autofocus example is shown in figure C.3. It should be noticed that, in general, autofocus algorithms require a number of iterations. This means that autofocus can be computationally very intensive. In [Bie94b] an autofocus algorithm is presented that can be implemented efficiently in hardware architecture. The algorithm search low resolution SAR images for high contrast areas. The computationally intensive mapdrift autofocusing is only performed on these small areas. The autofocusing parameters are then fed into the high resolution SAR imaging algorithm.

APPENDIX D

OVERLAP-ADD/DISCARD CONVOLUTION NETWORKS

In this appendix we show that the linear and circular convolution networks have two well-known equivalents in classical digital signal processing. Let us first analyze the linear convolution network. Recall equation (4.11) with $w \leftarrow z^N$

$$\hat{Y}(z;w)\Big|_{w \leftarrow z^N} = \sum_{m=0}^{2N-2}\sum_{n=0}^{N-1} a_n(z^N)x_{m-n}(z^N)z^{-m} \equiv \hat{Y}(z) \tag{D.1}$$

The coefficients $a_n(z^N)$ and $x_{m-n}(z^N)$ are the polyphase components of $A(z)$ and $X(z)$, i.e. they are order $N-1$ polynomials in z^N,

$$a_n(z^N) = \sum_{k=-\infty}^{\infty} a_{n+kN}z^{-kN}, \qquad x_{m-n}(z^N) = \sum_{k=-\infty}^{\infty} x_{m-n+kN}z^{-kN}$$

Substitution these expressions into equation (D.1) gives

$$\begin{aligned}\hat{Y}(z) &= \sum_{m=0}^{2N-2}\sum_{n=0}^{N-1}\left(\sum_{k=-\infty}^{\infty} a_{n+kN}z^{-kN}\right)\left(\sum_{k=-\infty}^{\infty} x_{m-n+kN}z^{-kN}\right)z^{-m} \\ &= \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty}\sum_{m=0}^{2N-2}\sum_{n=0}^{N-1} a_{n+lN}x_{m-n+(k-l)N}z^{-m}z^{-kN}\end{aligned} \tag{D.2}$$

The last step includes a rearrangement of the summations. We will now analyze equation (D.2) from the inner summation to the outer summation.

Obviously, the inner double summation is recognized as an order $2N-2$ polynomial in z. The $2N-1$ coefficients of this polynomial are recognized as the linear convolution result of

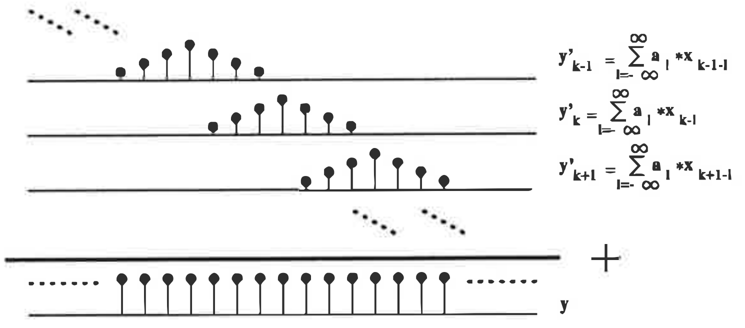


Figure D.1: The overlap-add method for construction the long linear convolution result **y** using short length $2N-1$ linear convolutions. In this example $N=4$.

two length N sequences with scalar elements a_{n+lN} and $x_{m-n+(k-l)N}$, $n=0,\cdots,N-1$. Let these length N sequences be defined as $\mathbf{a}_l$ and $\mathbf{x}_{k-l}$, then the inner double summation is the z-domain representation of the linear convolution result $\mathbf{a}_l * \mathbf{x}_{k-l}$.

The next observation is that the sequences $\mathbf{a}_l$ and $\mathbf{x}_{k-l}$ are in fact non-overlapping subsequences of the sequences $\mathbf{a} = \{a_j | j = \cdots,-1,0,1,\cdots\}$ and $\mathbf{x} = \{x_j | j = \cdots,-1,0,1,\cdots\}$. But then the inner triple summation is in fact the z-domain representation of the addition of the linear convolution results $\mathbf{a}_l * \mathbf{x}_{k-l}$ over l. Hence, the inner triple summation is still an order $2N-2$ polynomial in z, and thus the z-domain representation of a length $2N-1$ sequence with scalar elements. The latter statement implies that all convolution results $\mathbf{a}_l * \mathbf{x}_{k-l}$ with index k has a mutual zero delay factor.

Now let $\mathbf{y}'_k \triangleq \sum_{l=-\infty}^{\infty} \mathbf{a}_l * \mathbf{x}_{k-l}$ and let $Y'_k(z)$ be the z-domain representation of $\mathbf{y}'_k$. Then equation (D.2) can be simplified as

$$\hat{Y}(z) = \sum_{k=-\infty}^{\infty} Y'_k(z) z^{-kN} \tag{D.3}$$

Hence, in time domain equation (D.3) is nothing more adding the length $2N-1$ sequences $\mathbf{y}'_k$ with mutual factor N delay, as is shown in figure D.1 for the case that $N=4$. This mutual factor N delay causes an overlap of $N-1$ samples between subsequent sequences $\mathbf{y}'_k$ and $\mathbf{y}_{k+1}$. The linear convolution network can be described algorithmically, by introducing recursions in two dimensions. This description is shown in algorithm 3 in which we can recognize the overlap-add convolution method for solving a long linear convolution using multiple short linear convolutions [OS89].

Algorithm 3 : Overlap-Add

```
$\mathbf{a} \to \{\mathbf{a}_k\}, \quad \mathbf{x} \to \{\mathbf{x}_k\}$
for $all\ k$
    for $all\ l$
        $\mathbf{y}_k'^{(l)} = \mathbf{y}_k'^{(l-1)} + \mathbf{a}_l * \mathbf{x}_{k-l}$
    end
    $\mathbf{y}^{(k)} = \mathbf{y}^{(k-1)} + \mathbf{y}_k' * \delta_{kN}$
end
```

The same analysis can be made for the circular convolution network, and this will lead to the overlap-discard convolution method [OS89]. We will briefly give a description of the analysis similar to the linear convolution network, without going in too much details. Given the $2N-1$ polynomial coefficients $\widehat{x}_m(z^N)$ and the $N-1$ polynomial coefficients $a_n(z^N)$. Let $\widehat{\widehat{y}}_m(z^N)$ be their circular convolution results, given by equation (4.15)

$$\widehat{\widehat{y}}_m(z^N) = \sum_{n=0}^{N-1} a_n(z^N)\widehat{x}_{(m-n)\mathrm{mod}(2N-1)}(z^N)$$

Remind that for the circular convolution network we were only interested in $\widehat{\widehat{y}}_m(z^N)$ for $m = N-1, \cdots, 2N-2$. Observe that

$$(m-n)\mathrm{mod}(2N-1) = m-n \qquad \text{for} \quad \begin{array}{l} m = N-1, \cdots, 2N-1, \\ n = 0, \cdots, N-1 \end{array} \tag{D.4}$$

The z-domain representation is then

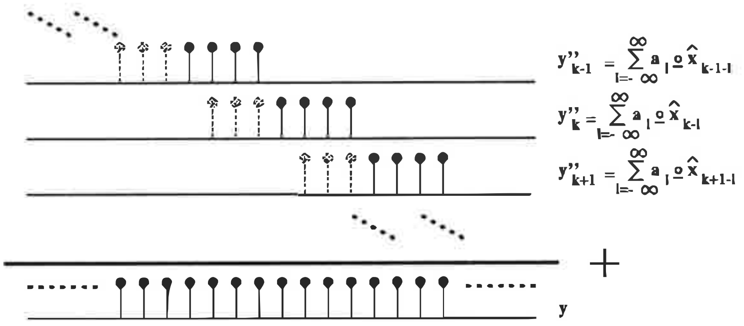
$$\widehat{\widehat{Y}}(z) = \sum_{m=N-1}^{2N-2} \sum_{n=0}^{N-1} a_n(z^N)\widehat{x}_{m-n}(z^N)z^{-m} \tag{D.5}$$

From equation (4.17) follows that

$$\widehat{x}_m(z^N) = \sum_{k=-\infty}^{\infty} x_{m+1+(k-1)N} z^{-kN} \tag{D.6}$$

Hence, equation (D.5) can be expressed

$$\widehat{\widehat{Y}}(z) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \sum_{m=N-1}^{2N-2} \sum_{n=0}^{N-1} a_{n+lN} x_{m-n+1+(k-l-1)N} z^{-m} z^{-kN} \tag{D.7}$$



$y''_{k-1} = \sum_{l=-\infty}^{\infty} a_l \underline{\circ} \hat{x}_{k-1-l}$
$y''_k = \sum_{l=-\infty}^{\infty} a_l \underline{\circ} \hat{x}_{k-l}$
$y''_{k+1} = \sum_{l=-\infty}^{\infty} a_l \underline{\circ} \hat{x}_{k+1-l}$
+
y

Figure D.2: The overlap-discard method for construction the long linear convolution result **y** using short length $2N-1$ circular convolutions. In this example $N = 4$.

Let $\hat{\mathbf{x}}_{k-l}$ be the length $2N-1$ sequence with scalar elements $x_{m-n+1+(k-l-1)N}$ for $m-n = 0, \cdots, 2N-2$. Introduce the combined circular convolution and discard operation $\underline{\circ}$. Then we can define the length N sequence $\mathbf{y}''_k$ as the summation over l of the last N element of the results of circularly convolving of $\mathbf{a}_l$ and $\hat{\mathbf{x}}_{k-l}$, given as $\mathbf{y}''_k \triangleq \mathbf{a}_l \underline{\circ} \hat{\mathbf{x}}_{k-l}$. The triple inner summation is then the z-transform of the subsequence of $\mathbf{y}''_k$. In fact, we have discarded the first $N-1$ samples. Thus in time domain equation (D.7) is nothing more than adding the length N sequences $\mathbf{y}''_k$ with mutual factor N delay, as is shown in figure D.2 for the case that $N = 4$. This description is shown in algorithm 4 in which we can recognize the overlap-discard convolution method for solving a long linear convolution using multiple short circular convolutions [OS89].

Algorithm 4 : Overlap-Discard

```
$\mathbf{a} \rightarrow \{\mathbf{a}_k\}$,   $\mathbf{x} \rightarrow \{\hat{\mathbf{x}}_k\}$
for $all\ k$
    for $all\ l$
        $\mathbf{y}''^{(l)}_k = \mathbf{y}''^{(l-1)}_k + \mathbf{a}_l \underline{\circ} \hat{\mathbf{x}}_{k-l}$
    end
    $\mathbf{y}^{(k)} = \mathbf{y}^{(k-1)} + \mathbf{y}''_k * \delta_{kN}$
end
```

BIBLIOGRAPHY

[AB74] R.C. Agarwal and C.S. Burrus. Fast One-Dimensional Digital Convolution by Multidimensional Techniques. *IEEE-ASSP*, ASSP-22(1):1–10, February 1974.

[AB75] R.C. Agarwal and C.S. Burrus. Number Theoretic Transforms to Implement Fast Digital Convolution. *Proceedings of the IEEE*, 63(4):550–560, April 1975.

[AC77] R.C. Agarwal and J.W. Cooley. New Algorithms for Digital Convolution. *IEEE-ASSP*, ASSP-25(5):392–410, October 1977.

[Alt93] Altera Corporation. *ALTERA Data Book*, August 1993.

[BBOW92] D. Blacknell, A.P. Blake, C.J. Oliver, and R.G. White. A Comparison of SAR Multilook Registration and Contrast Optimisation Autofocus Algorithms Applied to Real SAR Data. In *Proceedings Radar '92*, pages 363–366. IEE, 1992.

[BCM94] R. Bernadini, G.M. Cortelazzo, and G.A. Mian. A Sequential Multidimensional Cooley-Tukey Algorithm. *IEEE-SP*, SP-94(9):2430–2438, September 1994.

[BD74] M.G. Bellanger and J.L. Daguet. TDM-FDM Transmultiplexer: Digital Polyphase and FFT. *IEEE Transactions on Communications*, COM-22(9):1199–1205, September 1974.

[BD92] L.H.J. Bierens and E.F. Deprettere. A Matrix Description of Synthetic Aperture Radar Data. In J. Vandewalle, R. Boite, and M.Moonen, editors, *Proceedings EUSIPCO '92*, pages 103–106. Elseviers Science Publishers, 1992.

[BG88] W.D. Brown and D.C. Ghiglia. Some Methods for Reducing Propagation-induced Phase Errors in Coherent Imaging Systems I. Formalism. *Journal of the Optical Society of America A*, 5(6):924–941, June 1988.

[Bie93] L.H.J. Bierens. Real-Time SAR Processing Activities at TNO Physics and Electronics Laboratory. In *Proceedings IGARSS '93*, pages 1413–1415. IEEE, 1993.

[Bie94a] L.H.J. Bierens. A Feasibilty Study of an On-Board High Resolution Real-Time Airborne SAR Processor. In *Proceedings Microwaves '94*, pages 68–73. NEXUS, 1994.

[Bie94b] L.H.J. Bierens. A Subband Algorithm for Real-Time Autofocusing of SAR Images. In *Proceedings IGARSS '94*. IEEE, 1994.

[Bla85] R.E. Blahut. *Fast Algorithms for Digital Signal Processing.* Addison-Wesley, 1985.

[Boe95] M.J.G. Boerrigter. A Feasibility Study of a single Chip Implementation of Fast Convolution for Real Time SAR Processing. Master's thesis, University of Twente, August 1995.

[Bu90] J. Bu. *Systematic Design of Regulat VLSI Processor Arrays.* PhD thesis, Delft University of Technolgy, May 1990.

[BvHvB94] L.H.J. Bierens, R.M.E.M. van Heijster, and J.P. van Bezouwen. Data-Reduction Using Real-Time On-Board Sarellite SAR Processing. In *Proceedings Int. Workshop on Digital Signal Processing Techniques Applied to Space Communications*, pages 2.21–2.30. ESA, 1994.

[Cat92] Catalina Research Incorporated. *Data Sheet: CRV1M40 DSP board based on SHARP LH9124/LH9320 Chip Set*, October 1992.

[Cen88] A. Di Cenzo. A New Look at Nonseparable Synthetic Aperture Radar Processing. *IEEE Transactions on Aerospace and Electronic Systems*, AES-24(3):218–224, May 1988.

[CHV95] H. Chen, S. Van Huffel, and J. Vandewalle. Structured Total Least Norm and its Applcation to Parameter Estmation of Exponentially Damped Sinusoids. In *Proceedings Workshop on Circuits, Systems and Signal Procesing*, pages 45–50. ProRISC/IEEE Benelux, 1995.

[CM91] J.C. Curlander and R.N. McDonough. *Synthetic Aperture Radar, Systems and Signal Processing.* John Wiley & Sons, Inc., 1991.

[COM94] COMPASS. *Passport: Standard Cell Library 0.6 μ 3 V*, 1994.

[CPR89] C. Cafforio, C. Prati, and F Rocca. Synthetic Aperture Radar Focusing with Polyphase Filters. *Signal Processing*, 18:397–411, 1989.

[CPR91] C. Cafforio, C. Prati, and E. Rocca. SAR Data Focusing Using Seismic Migration Techniques. *IEEE Transactions on Aerospace and Electronic Systems*, AES-27(2):194–207, March 1991.

[CR83] R.E. Crochiere and L.R. Rabiner. *Multirate Digital Signal Processing.* Prentice-Hall, 1983.

[Cut70] L.J. Cutrona. Synthetic Aperture Radar. In M.I. Skolnik, editor, *Radar Handbook*, chapter 23. McGraw-Hill, 1970.

[CVLH61] L.J. Cutrona, W.E. Vivian, E.N. Leith, and G.O. Hall. A High-Resolution Radar Combat-Surveillance System. *IRE Transactions on Military Electronics*, MIL-5:127–131, April 1961.

[Del70] G.W. Deley. Waveform Design. In M.I. Skolnik, editor, *Radar Handbook*, chapter 3. McGraw-Hill, 1970.

[Dep93] E.F. Deprettere. Example of Combined Algorithm Development and Architecture Design. *Integration, the VLSI Journal*, 16(3):199–220, 1993.

[DHW93] E.F. Deprettere, P. Held, and P. Wielage. Model and Methods for Regular Array Design. *Int. Journal of High Speed Electronics*, 4(4), 1993.

[DJCM92] J. Dall, J.H. Jorgensen, E.L. Christensen, and S.N. Madsen. Real-Time Processor for the Danish Airborne SAR. *IEE Proceedings-F*, 139(2):115–121, April 1992.

[EGCJ89] P.H. Eichel, D.C. Ghiglia, and Jr. C.V. Jakowatz. Speckle Processing Method for Synthetic Aperture Radar Phase Correction. *Optics Letters*, 14(1):1–3, January 1989.

[ES92] G.P.M. Egelmeers and P.C.W. Sommen. Relation between Reduced Dimension Time and Frequency Domain Adaptive Algorithm. In J. Vandewalle, R. Boite, and M.Moonen, editors, *Proceedings EUSIPCO '92*, pages 1065–1068. Elseviers Science Publishers, 1992.

[ESA92] ESA. *ESA ERS-1 Product Specification*, June 1992. No. ESA SP-1149.

[Fel93] T. Felhauser. Digital Signal Processing for Optimum Wideband Channel Estimation in the Presence of Noise. *IEE Proceedings-F*, 140(3):179–186, June 1993.

[FS90] G. Franceschetti and G. Schirinzi. A SAR Processor Based on Two-Dimensional FFT Codes. *IEEE on Aerospace and Electronics Systems*, AES-26(2):356–366, March 1990.

[FW85] I.P. Finley and J.W. Wood. An Investigation of Synthetic Aperture Radar Autofocus. Internal RSRE Memorandum 3790, Royal Signals & Radar Establishment, April 1985.

[GB88] D.C. Ghiglia and W.D. Brown. Some Methods for Reducing Propagation-induced Phase Errors in Coherent Imaging Systems II. Numerical Results. *Journal of the Optical Society of America A*, 5(6):942–957, June 1988.

[GEC93a] GEC Plessey. *Digital Video and Digital Signal Processing, IC Handbook*, December 1993. Data Sheet: PDSP16510A, Stand Alone FFT Processor.

[GEC93b] GEC Plessey. *Digital Video and Digital Signal Processing, IC Handbook*, December 1993. Data Sheet: PDSP16116/A, 16 by 16 bit complex multiplier.

[GS58] U. Grenander and G. Szego. *Toeplitz Forms and their Applications*. University of California Press, 1958.

[GV92] A. Gilloire and M. Vetterli. Adaptive Filtering in Subbands with Critical Sampling: Analysis, Experiments, and Applications to Acoustic Echo Cancellation. *IEEE Transactions on Signal Processing*, 40(8):1862–1875, August 1992.

[GvL89] G.H Golub and C.F. van Loan. *Matrix Computations*. The John Hopkins University Press, 1989.

[Har70] R.O. Harger. *Synthetic Aperture Radar Systems*. Academic Press, 1970.

[Hov80] S.A. Hovanessian. *Introduction to Synthetic Array and Antenna Imaging Radars*. Arthech House, 1980.

[HSKP92] P. Hoogenboom, P. Snoeij, P.J. Koomen, and H. Pouwels. The PHARUS Project, Results of the Definition Study Including the SAR Testbed PHARS. *IEEE Transactions on Geoscience and Remote Sensing*, GE-30(4):723–735, July 1992.

[Jin86] M.Y. Jin. Optimum Doppler Centroid Estimation for SAR Data from a Quasi-Homogeneous Source. *IEEE Transactions on Geoscience and Remote Sensing*, GE-24(6):1022–1025, November 1986.

[Kun88] S.Y. Kung. *VLSI Array Processors*. Prentice-Hall, Inc., 1988.

[LHCW85] F.K. Li, D.H. Held, J.C. Curlander, and C. Wu. Doppler Parameter Estimation for Spaceborne Synthetic-Aperture Radars. *IEEE Transactions on Geoscience and Remote Sensing*, GE-23(1):47–55, January 1985.

[Mad89] S.N. Madsen. Estimating the Doppler Centroid of SAR Data. *IEEE Transactions on Aerospace and Electronic Systems*, AES-25(2):134–140, March 1989.

[McW92] J.G. McWirther. Algorithmic Engineering in Adaptive Signal Procesing. *IEE Proceedings-F*, 139(3):226–232, June 1992.

[Mer95] Mercury Computer Systems Inc. *Realtime Synthetic Aperture Radar Processing on the RACE Multicomputer*, 1995. Application Note 203.0.

[Mor90] J.R. Moreira. A New Method of Aircraft Motion Error Extraction From Raw Data for Real Time Motion Compensation. *IEEE Transactions on Geoscience and Remote Sensing*, GE-28(4):620–626, July 1990.

[Mor92] A. Moreira. Real-Time Synthetic Aperture Radar (SAR) Processing with a New Subaperture Approach. *IEEE Transactions on Geoscience and Remote Sensing*, GE-30(4):714–722, July 1992.

[MP92] J.G. McWirther and I.K. Proudler. Algorithmic Engineering: A Worked Example. In *Proceedings EUSIPCO '92*, pages 5–12. Elseviers Science Publishers B.V., 1992.

[Nor63] D.O. North. An Analysis of the Factors which Determine Signal/Noise Discrimination in Pulsed-Carrier Systems. *Proceedings of the IEEE*, 51:1016–1027, July 1963.

[Oli89] C.J. Oliver. Review Article: Synthetic Aperture Radar Imaging. *Journal of Applied Physics*, 22(4):871–890, 1989.

[OS89] A.L. Oppenheim and R.W. Schafer. *Discrete-Time Signal Processing*. Prentice-Hall, Inc., 1989.

[Ott89] M.P.G. Otten. Phase Error Estimation for PHARUS: A Review of Autofocus and Related Algorithms. Internal Report FEL-89-C243, TNO Physics and Electronics Laboratory, September 1989.

[Ott91] M.P.G. Otten. Phase Error Estimation for PHARUS: theory and Results of Autofocus and Doppler Centroid Estimation. Internal Report FEL-91-C185, TNO Physics and Electronics Laboratory, August 1991.

[Ott94] M.P.G. Otten. Stand Pharus-gondel, PHARUS doc.no. 22340.W24.AS04. TNO Physics and Electronics Laboratory, The Hague, The Netherlands, May 1994.

[Pap84] A. Papoulis. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, 1984.

[Por80] R. Portnoff. Time-Frequency Representation of Digital Signals and Systems based on Short-Time Fourier Analysis. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-28(1):55–69, February 1980.

[PV95] S.M. Phoong and P.P. Vaidyanathan. One- and Two-Level Filter-Bank Convolvers. *IEEE Transactions on Signal Processing*, ASSP-43(1):116–133, january 1995.

[Rad72] C.M. Rader. Discrete Convolutions via Mersenne Transforms. *IEEE Transactions on Computers*, C-21(12):1296–1273, December 1972.

[RRB$^+$94] R.K. Raney, H. Runge, R. Bamler, I.G. Cumming, and F.H. Wong. Precision SAR Processing Using Chirp Scaling. *IEEE Transactions on Geoscience and Remote Sensing*, GE-32(4):786–799, July 1994.

[Sch91] L.L. Scharf. *Statistical Signal Processing*. Addison-Wesley, 1991.

[Sin67] R.C. Singleton. A Method for Computing the Fast Fourier Transform with Auxiliary Memory and Limited High Speed Storage. *IEEE Transactions on Audio and Electroacoustics*, AU-15(2):91–98, June 1967.

[Sko70] M.I. Skolnik, editor. *Radar Handbook*. McGraw-Hill, 1970.

[Sko85] M.I. Skolnik. *Introduction to Radar Systems*. McGraw-Hill, 1985.

[SR62] C.W. Sherwin and R.D. Rawcliffe. Some Early Developments in Synthetic Aperture Radar Systems. *IRE Transactions on Military Electronics*, MIL-6:111–115, April 1962.

[Ste91] A. Steffen. *Digital Pulse Compresion Using Multirate Filter Banks*. Hartung-Gorre, 1991.

[Sto71] H.S. Stone. Parallel Processing with the Perfect Shuffle. *IEEE Transactions on Computers*, C-20(2):153–161, Februari 1971.

[SV86] K. Swaminathan and P.P. Vaidyanathan. Theory and Design of Uniform DFT, Parallel, Quadrature Mirror Filter Banks. *IEEE Transactions on Circuits and Systems*, CAS-33(12):1170–1191, December 1986.

[UMF82] F.T. Ulaby, R.K. Moore, and A.K. Fung. *Microwave Remote Sensing, Active and Passive, Volume II: Radar Remote Sensing Fundamentals and Emission Theory.* Addison-Wesley, 1982.

[Vai90] P.P. Vaidyanathan. Multirate Digital Filter, Filter Banks, Polyphase Networks, and Applications: A Tutorial. *Proceedings of the IEEE*, 78(1):56–93, January 1990.

[Vai93] P.P. Vaidyanathan. Orthonormal and Biorthonormal Filter Banks as Convolvers, and Convolutional Coding Gain. *IEEE Transactions on Signal Processing*, ASSP-41(6):2110–2130, June 1993.

[Vet86] M. Vetterli. Filter Banks Allowing Perfect Reconstruction. *Signal Processing*, 10:219–244, April 1986.

[Vet87] M. Vetterli. Theory of Multirate Filter Banks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35(3):356–372, March 1987.

[Vet88] M. Vetterli. Running FIR and IIR Filtering Using Multirate Filter Banks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-36(5):730–738, May 1988.

[vHDvB$^+$94] A.W.P. van Heijningen, B. Dondertman, R. v.d. Bos, C. van 't Wout, R. Prevo, C.D. Pieterse, and A.A. Romein. Testbedomgeving Real Time SAR. Internal Report FEL-94-A292, TNO Physics and Electronics Laboratory, May 1994.

[Wei69] C.J. Weinstein. Roundoff Noise in Floating Point Fast Fourier Transform Computation. *IEEE Transactions on Audio and Electroacoustics*, AU-17(3):209–215, September 1969.

[WLJ82] C. Wu, K.Y. Liu, and M. Jin. Modeling and a Correlation Algorithm for Spaceborne SAR Signals. *IEEE Transactions on Aerospace and Electronic Systems*, AES-18(5):563–575, September 1982.

SAMENVATTING

In dit proefschrift hebben we een schematische ontwerpmethodiek geïntroduceerd waarmee efficiënte hardware architecturen kunnen worden ontworpen ten behoeve van multirate convolutiesystemen. Deze methodiek is ook toepasbaar op andere multirate signaalbewerkingssystemen. De ontwerpmethodiek gaat uit van een grafische beschrijving van een algoritme in plaats van een algebraïsche beschrijving. Het voordeel hiervan is dat een grafische beschrijving zich veel makkelijker laat manipuleren dan een algebraïsche beschrijving, vooral wanneer de laatste volledig in detail worden uitgeschreven. Het is bekend dat multirate filterbanken en convolutiesystemen veel overeenkomsten vertonen. Door middel van de grafische manipulatie die we hebben ontwikkeld zijn deze overeenkomsten op een eenvoudige en illustratieve wijze expliciet gemaakt. Bovendien kan een multirate convolutie-algoritme met behulp van deze grafische ontwerpmethodiek direct in een prototype architectuur of in dedicated VLSI processors worden geïmplementeerd.

In veel complexe real-time signaalbewerkings-toepassingen als radar, sonar en medical imaging, is behoefte aan een schematische ontwerpmethodiek die rekening houdt met typische toepassings-specifieke beperkingen als verwerkingssnelheid, processoromvang en geheugenbeheer. Het uitgangspunt van onze ontwerpmethodiek is dat complexe signaalbewerkingsalgoritmes opgedeeld worden in kleinere, minder complexe sub-algoritmes. Uiteindelijk leidt deze aanpak tot winst in ontwerptijd en reduceert het de ontwerprisico's en complexiteit.

We hebben de bruikbaarheid van onze ontwerpmethodiek aangetoond in het ontwerp van een multirate convolutiesysteem, toegepast in de range-compressie en azimutcompressie ten behoeve van real-time SAR processing. Range-compressie en azimutcompressie zijn de kritische onderdelen van een real-time SAR processor, met name als beperkte omvang, laag vermogensverbruik en hoge verwerkingssnelheid de ontwerpeisen zijn. Het toepassen van

onze ontwerpmethodiek heeft geresulteerd in een efficiënte VLSI convolutie-processor. Naar verwachting zal een real-time on-board SAR processor gebaseerd op deze convolutie-processor tot een factor 10 kleiner zijn (zowel qua omvang als vermogensverbruik) dan een SAR processor gebaseerd op commercieel verkrijgbare DSP-kaarten, zie onder meer [Cat92, Mer95].

Onze VLSI convolutie-processor is veel krachtiger dan een DSP-kaart implementatie, als we de performance van een convolutie-processor uitdrukken in termen van verwerkingssnelheid per eenheid van processoromvang en per eenheid van vermogensverbruik. Een grove schatting is dat de verwerkingssnelheid van onze convolutie-chip vergelijkbaar is met de verwerkingssnelheid van één state-of-the-art DSP-kaart. De omvang en het vermogensverbruik zijn echter veel kleiner, en dat ondanks het feit dat onze VLSI convolutie-processor ca. twee keer zo veel bewerkingen uitvoert. Bovendien hebben we laten zien dat onze VLSI architectuur voldoet aan de stringente radar signaalverwerkings-specificaties ten aanzien van nauwkeurigheid en dynamisch bereik door middel van het gebruik van een hybride floating point formaat. Met een DSP-kaart oplossing is het in het algemeen moeilijk om aan deze specificaties te voldoen doordat deze vaak het block-floating point formaat gebruiken.

Dit geeft ons de motivatie om te stellen dat ontwerpers van hardware-architecturen de kosten van een DSP architectuur niet horen uit te drukken in termen van complexiteit, maar in termen van *overall performance*. Hieronder vallen bijvoorbeeld verwerkingssnelheid, processoromvang, vermogensverbruik, nauwkeurigheid en dynamisch bereik. De exacte definitie van overall performance wordt natuurlijk bepaald door de toepassing; het is niet zinvol om bijvoorbeeld een SAR processor te implementeren in dedicated VLSI hardware als slechts off-line verwerking vereist is. In de inleiding hebben we een aantal real-time SAR toepassingen genoemd (verkenning, SAR data transmissie, fixed/moving target indication en on-board systeem monitoring) die wenselijk zouden zijn aan boord van kleine platformen zoals straaljagers en unmanned airborne vehicles. Deze zullen in de nabije toekomst mogelijk zijn als dedicated VLSI architecturen (en vooral de convolutie-architectuur) worden gebruikt. Deze lijst kan worden aangevuld met bijvoorbeeld toepassingen voor SAR processing aan boord van een satelliet.

CURRICULUM VITAE

Laurens Bierens was born in Februari 2, 1966 in Amsterdam, Netherlands. After graduation the secondary school "De Rietlanden" in Lelystad in 1984, he started his study Electrical Engineering at the Delft University of Technology. During his study, in the years 1986/1987, he was a member of the board of the "Electrotechnische Vereenigning", the study association of the faculty of Electrical Engineering. In 1990 he finished his study at the Network Theory Group headed by Prof. Patrick Dewilde. During his graduation research, which was in the field of hardware architecture design for computer graphics, he was supervised by Dr. Ed Deprettere. He finished his study in 1990, after which he joined TNO Physics and Electronics Laboratory (TNO-FEL), The Hague, Netherlands, as a research engineer. His main work at TNO-FEL has been in the design of dedicated architectures for real-time SAR processing. He has served as a project manager in several projects related to real-time SAR processing. The main results of his research activities are presented in this thesis. In the future he will continue to do research in the field of real-time SAR processing at TNO-FEL. The emphasis will be more on real-time SAR applications and on the design of VLSI architectures.