Digital Signal Processing: A Computer Science Perspective
Jonathan Y. Stein

Print ISBN 0-471-29546-9 Online ISBN 0-471-20059-X

Digital Signal Processing

Digital Signal Processing

A Computer Science Perspective

Jonathan (Y) Stein

A Wiley-Interscience Publication
JOHN WILEY & SONS, INC.
New York • Chichester • Weinheim • Brisbane • Singapore • Toronto

ISBN 0-471-20059-X.

This title is also available in print as ISBN 0-471-29546-9

For more information about Wiley products, visit our web site at www.Wiley.com.

Library of Congress Cataloging in Publication Data

Stein, Jonathan Y.
Digital signal processing: a computer science perspective/Jonathan Y. Stein
p. cm.
"A Wiley-Interscience publication."
Includes bibliographical references and index.
ISBN 0-471-29546-9 (cloth: alk. paper)
1. Signal processing — Digital techniques. I. Title.

TK5102.9. S745 2000
621.382′2 — dc21 00-035905

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

To Ethel, Hanna and Noga

Contents

Part IV Applications

Preface

I know what you are asking yourself—'there are a lot of books available about DSP, is this book the one for *me*?' Well that depends on who *you* are. If

- you are interested in doing research and development in one of the many state-of-the-art applications of DSP, such as speech compression, speech recognition, or modem design,
- your main proficiency is in computer science, abstract mathematics, or science rather than electronics or electrical engineering,
- your mathematical background is relatively strong (flip back now to the appendix—you should be comfortable with about half of what you see there),

then you are definitely in the *target group* of this book. If in addition

- you don't mind a challenge and maybe even enjoy tackling brain-teasers,
- you're looking for one comprehensive text in all aspects of DSP (even if you don't intend reading all of it now) and don't want to have to study several different books with inconsistent notations, in order to become competent in the subject,
- you enjoy and learn more from texts with a light style (such as have become common for computer science texts) rather than formal, dry tomes that introduce principles and thereafter endlessly derive corollaries thereof,

then this is probably the book you have been waiting for.

This book is the direct result of a chain of events, the first link of which took place in mid-1995. I had been working at a high-tech company in Tel

Aviv that was a subsidiary of a New York company. In Tel Aviv it was relatively easy to locate and hire people knowledgeable in all aspects of DSP, including speech processing, digital communications, biomedical applications, and digital signal processor programming. Then, in 1995, I relocated to a different subsidiary of the same company, located on Long Island, New York. One of my first priorities was to locate and hire competent DSP software personnel, for work on speech and modem signal processing.

A year-long search turned up next to no-one. Assignment agencies were uncertain as to what *DSP* was, advertisements in major New York area newspapers brought irrelevant responses (digital design engineers, database programmers), and, for some inexplicable reason, attempts to persuade more appropriate people from Silicon Valley to leave the California climate, for one of the worst winters New York has ever seen, failed.

It struck me as rather odd that there was no indigenous DSP population to speak of, in an area noted for its multitude of universities and diversity of high-tech industries. I soon found out that DSP was not taught at undergraduate level at the local universities, and that even graduate-level courses were not universally available. Courses that *were* offered were Electrical Engineering courses, with Computer Science students never learning about the subject at all. Since I was searching for people with algorithm development and coding experience, preferably strong enough in software engineering to be able to work on large, complex software systems, CS graduates seemed to be more appropriate than EEs. The ideal candidate would be knowledgeable in DSP and would in the *target group* mentioned above.

Soon after my move to New York I had started teaching graduate level courses, in Artificial Intelligence and Neural Networks, at the Computer and Informations Sciences department of Polytechnic University. I inquired of the department head as to why a DSP course was not offered to Computer Science undergraduates (it *was* being offered as an elective to Electrical Engineering graduate students). He replied that the main reason was lack of a suitable teacher, a deficiency that could be easily remedied by my volunteering.

I thus found myself 'volunteered' to teach a new Computer Science undergraduate elective course in DSP. My first task was to decide on course goals and to flesh out a syllabus. It was clear to me that there would be little overlap between the CS undergraduate course and the EE graduate-level course. I tried to visualize the ideal candidate for the positions I needed to fill at my company, and set the course objectives in order to train the perfect candidate. The objectives were thus:

- to give the student a basic understanding of the theory and practice of DSP, at a level sufficient for reading journal articles and conference papers,
- to cover the fundamental algorithms and structures used in DSP computation, in order to enable the student to correctly design and efficiently code DSP applications in a high-level language,
- to explain the principles of digital signal processors and the differences between them and conventional CPUs, laying the framework for the later in-depth study of assembly languages of specific processors,
- to review the background and special algorithms used in several important areas of state-of-the-art DSP research and development, including speech compression/recognition, and digital communications,
- to enable the student who completes the course to easily fit in and contribute to a high-tech R&D team.

Objectives defined, the next task was to choose a textbook for the course. I perused web sites, visited libraries, spoke with publisher representatives at conferences, and ordered new books. I discovered that the extant DSP texts fall into three, almost mutually exclusive, categories.

About 75% of the available texts target the EE student. These books assume familiarity with advanced calculus (including complex variables and ordinary differential equations), linear system theory, and perhaps even stochastic processes. The major part of such a text deals with semirigorous proofs of theorems, and the flavor and terminology of these texts would certainly completely alienate most of my target group. The CS student, for example, has a good basic understanding of derivatives and integrals, knows a little linear algebra and probably a bit of probability, but has little need for long, involved proofs, is singularly uninterested in poles in the complex plane, and is apt to view too many integral signs as just so many snakes, and flee in terror from them.

In addition, these *type-one* texts ignore those very aspects of the subject that most interest our target students, namely algorithm design, computational efficiency and special computational architectures, and advanced applications. The MAC instruction and Harvard architecture, arguably the defining features of digital signal processors, are generally not even mentioned in passing. Generally only the FFT, and perhaps the Levinson-Durbin recursion, are presented as algorithms, and even here the terminology is often alien to the CS student's ear, with no attention paid to their relation with other problems well known to the computer scientist. The exercises generally involve extending proofs or dealing with simplistic signals that can be handled analytically; computer assignments are rare.

Finally, due perhaps to the depth of their coverage, the *type-one* texts tend to cover only the most basic theory, and no applications. In other words, these books finish before getting to the really interesting topics. Some cover the rudiments of speech processing, e.g. LPC and cepstral coefficients, but all consider speech compression and modem design beyond their scope. More advanced or specific texts are thus absolutely necessary before real-world applications can be tackled. These texts thus do not achieve our goal of preparing the student for participation in a real R&D team.

The next category, counting for about 20% of the texts, *do* target people who are more at home with the computer. *Type-two* texts tend to be 'recipe books', often accompanied by a diskette or CD. The newer trend is to replace the book with interactive instruction and experimentation software. These books usually contain between fifty and one hundred black box routines that can be called from a high-level language (e.g. C or MATLAB). The bulk of the text consists of instructions for calling these routines, with discussion of the underlying theory kept to a minimum.

While very useful for the computer professional who on odd occasions has need for some DSP procedures, these books do not instill a deep unified comprehension of the subject. Admittedly these books often explain algorithms in greater depth than *type-one* texts, but our target readers would benefit even more from a combination of *type-one* depth with *type-two* emphasis.

Of course there is nothing wrong with obtaining a well tested program or routine that fulfills the purpose at hand. Indeed it would not be prudent for the implementor to reinvent wheels in places where tire shops abound. However, due to their generality, library routines are often inefficient and may even be impractical for specific purposes. I wanted to enable my students to meet specific DSP needs by evaluating existing programs and library routines, or by writing original, tailored DSP code as required. The reader should also be able to port libraries to a new platform, understanding both the algorithm and the platform idiosyncrasies.

Finally, there are *type-three* texts, often written by DSP processor manufacturers. They emphasize the architecture, programming language, and programming tools of the manufacturer's line of digital signal processors, and while they may explain some theory, they mostly assume prior knowledge or claim that such knowledge is not really required for the comprehension of the subject matter. The programming techniques developed, usually in lengthy detail, may be applicable to some extent to other manufacturers' processors, but considerable adaptation would normally be required. *Type-three* texts tend to stress FIR and IIR filter structures, the radix 2 FFT algorithms, the LMS and perhaps Viterbi algorithms, and often describe various practical applications of these in great depth.

Due to their lack of mathematical sophistication, these books do not attempt to seriously treat DSP theory. Such critical topics as the sampling theorem, filtering, and adaptive systems are only trivially covered; true explanation of noise, filtering, and Fourier transforms are replaced by historical accounts, and algorithms are displayed in pseudocode *fait accompli* rather than derived. On the other hand, the manufacturers apparently feel that the typical reader will be lacking in CS background, and thus overly stress such obvious features as loops and numeric representation.

I thus reached the conclusion that none of the available DSP texts was truly suitable for the course, and was compelled to create my own course materials. These became the corner-stone of the present book. Often I found myself rethinking my own understanding of the subject matter, and frequently connections with other computer science subjects would only become clear during lecture preparation, or even during the lecture itself. I also found that the elimination of the conventional mathematical apparatus and rigorous proofs not only did not deplete the subject matter of meaning, but actually enriched it.

The topics included in this text may, at first, surprise the reader who is used to more conventional DSP texts. Subjects such as the matched filters, adaptive algorithms, the CORDIC algorithm, the Viterbi algorithm, speech compression, and modern modem theory are normally considered too complex and specialized for presentation at this level. I have found that these *advanced* topics are no more difficult for the newcomer to grasp than filter design or limit cycles, and perhaps more interesting and relevant. However, in order to keep the book size moderate, some of the more classical subjects had to be curtailed. These subjects are adequately covered in traditional texts, which may be consulted to supplement the present one.

Even so, the present book contains more material than can be actually taught in a single-semester course. A first course in DSP could cover most of the material in the early chapters, with the instructor then selecting algorithms and applications according to personal preference. The remaining subjects may be relegated to a more advanced course, or be assigned as self-study topics. My initial course went through the basic theory at break-neck speed, in order to rapidly get to speech compression and recognition. A second attempt emphasized modems and DSP for data communications.

Every section ends with a number of exercises that are designed to be entertaining and enriching. Some of these should not be difficult for the reader who understands the section, being designed to reinforce basic understanding of the material. Many are somewhat challenging, complementing the text, extending the theory, or presenting actual applications of the subject studied. Some are only loosely defined; for these one can give a quick answer, or develop them into a term project. Others introduce new material that will ease the understanding of the following sections, as well as widening the reader's DSP horizons.

I purposely avoid taking sides on the divisive issue of programming language and environment for algorithm design and test on general-purpose computers. Realizing that C, MATLAB, SPW, Mathematica and the like will all have their staunch supporters, and all have their strengths and weaknesses, I leave it to the student or instructor to select that language with which they are the most comfortable. Every seasoned programmer is most effective in his or her native language, and although some languages are obviously better DSP 'environments' than others, the difference can be minimized by the use of appropriate libraries.

Although the book was written to serve as a course textbook, it may be used by non-students as well. DSP practitioners are like master craftsmen; when they are called upon to construct some object they must exploit their box of tools. Novices have only a few such tools, and even these may not be sufficiently sharp. With time more tools are acquired, but almost all craftsmen tend to continue using those tools with which they have the most experience. The purpose of this book is to fill the toolbox with tools, and to help the DSP professional become more proficient in their proper use. Even people working in the field several years will probably find here new tools and new ways of using tools already acquired.

I would like to thank my students, who had to suffer through courses with no textbook and with continually changing syllabus, for their comments; my colleagues, particularly Yair Karelic, Mauro Caputi, and Tony Grgas, for their conscientious proofreading and insights; my wife Ethel for her encouragement (even allowing me untold late-night sessions banging away at the keyboard, although she had long ago banished all computers from the house); and our two girls, Hanna and Noga, who (now that this book is complete) will have their father back.

Jonathan (Y) Stein
Jerusalem, Israel
31 December 1999

Digital Signal Processing: A Computer Science Perspective
Jonathan Y. Stein

Print ISBN 0-471-29546-9 Online ISBN 0-471-20059-X

Index

page numbers of references to:
- definitions are underlined
- exercises are in *italics*