INCLUDES

FREE
NEWNES ONLINE
MEMBERSHIP

Newnes

DIGITAL SIGNAL PROCESSING

WORLD CLASS DESIGNS

- Hand-picked content selected by Kenton Williston, Editor of DSP DesignLine
- Proven best design practices for image, audio, and video processing
- Case histories and design examples get you off and running on your current project

Kenton Williston EDITOR

TN911.72
D574.2

Digital Signal Processing World Class Designs

Kenton Williston

with

Rick Gentile
Keith Jack
David Katz
Nasser Kehtarnavaz
Walt Kester
Dake Liu
Robert Meddins
Robert Oshana
Ian Poole
Khalid Sayood
Li Tan

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Newnes is an imprint of Elsevier

ELSEVIER

Newnes

Newnes is an imprint of Elsevier
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA
Linacre House, Jordan Hill, Oxford OX2 8DP, UK

Library of Congress Cataloging-in-Publication Data
Application submitted.

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

ISBN: 978-1-85617-623-1

09 10 11 10 9 8 7 6 5 4 3 2 1

Printed in the United States of America

Digital Signal Processing World Class Designs

Preface

What is this DSP thing I keep hearing about? What's so great about it? What do DSP engineers do? And how do I get in on the action? Great questions! Take a journey with me and all will be revealed...

Defining DSP

First, let's figure out what DSP is. The acronym DSP has two alternate meanings: Digital Signal Processing and Digital Signal Processor. Let's look at each, starting with digital signal processing. The meaning of "digital" is obvious—it means we are working in the world of 1's and 0's, and not in the analog world. The idea of a "signal" is a bit trickier. Our friend Wikipedia defines the term as "any time-varying or spatial-varying quantity." Speech is an example of a time-varying quantity; the pitch and volume of a voice changes from one moment to the next. A photograph is an example of a space-varying quantity; the color and brightness of an image are different in different areas of the photo. Now we are left to define "processing." This is a broad concept, but it generally involves analysis and manipulation using mathematical algorithms. For example, we could analyze a voice recording to determine its pitch, and we could manipulate a photograph by adjusting its colors.

DSP applications fall into four main categories:

- Communications
- Audio, video, and imaging (sometimes referred to as media processing)
- Motion control
- Military and aerospace

Of these areas, communications and video receive the most attention. Both areas are evolving rapidly, and both impose high computational loads. In addition, both areas include systems with severely limited power budgets.

With this background, it's easy to define the term digital signal processor. A digital signal processor is a simply a processor with specialized features for signal processing. For example, many signal processing algorithms involve multiplication followed by addition, an operation commonly referred to as a multiply accumulate or MAC. Since the MAC operation is so common in signal processing, all digital signal processors include special MAC hardware. As another example. many DSP applications have limited power budgets. Thus, many DSPs offer advanced power-saving features, such as the ability to change speeds and voltages n the fly—a feature known as dynamic frequency and voltage scaling.

It is important to note that digital signal *processing* does not require digital signal *processors*. Many signal processing systems use general-purpose processors (such as those available from ARM and MIPS) or custom hardware (built using ASICs and FPGAs). Many systems use a mix of hardware. For example, many systems contain both a DSP and a general-purpose processor (GPP).

The Role of the DSP Engineer

So much for the basics. What do DSP engineers actually do? My friend Shiv Balakrishnan recently wrote an article on this topic, and I agree with his conclusion that DSP engineers typically fall into three categories:

1. System designers create algorithms and (in some cases) the overall system.
2. Hardware designers implement (1) in hardware.
3. Programmers implement (1) in software, either by using hardware created by (2) or by using off-the-shelf hardware.

This book is intended mainly for programmers. This is the most common role for DSP engineers, and the role that is easiest to address without getting into graduate-level concepts. Nonetheless, it is worth looking at each of these jobs in detail, and exploring the skill sets for each.

System Designer

The system designer focuses on the big picture. This engineer may design the overall functionality of the system, including all of the attendant algorithms, or they may focus on specific subsystems and algorithms. The latter case is common for products such as cell phones where the system complexity is too great for a single engineer.

The system designer is referred to as a domain expert because they need an expert understanding of the system requirements and how to meet them. This includes expertise on the analog world, because most DSP systems have analog inputs and outputs. For example, wireless system designers must know how signals degrade as they propagate through the air.

The system designer also needs top-notch algorithmic expertise. To meet these demands, the system designer often needs a master's degree or even a PhD.

The system designer builds a functional model of the system using graphical tools such as *Simulink* and *LabVIEW* as well as text-based tools like MATLAB and C. The system designer often does not get into the details of the hardware and software design, but they must understand the basics of these disciplines. Even the most brilliant design is worthless if no one can build it! System designers must also take great care to ensure that hardware designers and programmers fully understand the functional model. Among other things, this means that the system designer must provide a means of testing the hardware and software against the system model. To meet these goals, system designers are increasingly turning to Electronic System Level (ESL) tools. ESL tools perform a number of functions, including automatic generation of reference hardware and software, as well as generation of test vectors.

Hardware Designer

Once the system is designed, it's the hardware designer's turn. The role of the hardware designer varies widely. As with the system designer, the hardware designer may work on the entire system or may focus on specific subsystems. The hardware designer may create custom hardware, or they may build a system using off-the-shelf parts. For custom hardware, designers once turned to *ASICs*, but ASIC design has become prohibitively expensive for all but the highest-volume products. As a result, hardware designers often use *FPGAs* instead. *Structured ASICs* are also an option, particularly for medium-volume applications.

In any of these cases, the hardware designer realizes the hardware as a set of blocks. Traditionally, the designer would implement each block in hand-coded *RTL* (either *VHDL* or *Verilog*), verify it, and optimize it. While hand-coded RTL is still in use, hardware engineers increasingly rely on ESL tools to generate hardware. This is particularly true for key DSP algorithms like FFTs and FIR filters. ESL tools have become quite proficient at generating hardware for these algorithms. In addition to the custom-built blocks, most applications also include one or more programmable processors. The processors may be implemented inside an ASIC or FPGA, or the hardware designer may use off-the shelf processors.

For systems that use off-the-shelf hardware, the hardware designer's job is much simpler. However, the hardware designer still has to make many careful choices in order to meet the system requirements. The processors must have enough performance to handle the workload, the buses must have enough bandwidth to handle the data, and so on. In many cases, it is impossible to find off-the-shelf hardware that fully meets the system requirements, so the hardware designer must create a small amount of custom hardware.

For example, many systems use an FPGA to implement I/O that is not available in the off-the-shelf processor.

The upshot of all of this is that hardware designers need a variety of skills. Although ESL tools are lightening the workload, the hardware designer must have good RTL coding skills. Obviously, hardware designers must understand the algorithms they implement. They must also understand the requirements of the software so they can make wise design decisions. Although it is possible to meet all of these requirements with a bachelor's-level education, a master's degree is quite helpful.

Programmer

Next up: the programmer. The programmer writes code to implement the remaining functionality on the systems processor or processors. Like programmers in any other field, the DSP programmer generally writes in C/C++. However, DSP code is unusual for two reasons. First, DSP code often begins as a MATLAB or Simulink model. Thus, tools that convert *MATLAB to C* are drawing a great deal of interest. Second, DSP code requires heavy optimization for performance, size, and power. In many cases, this requires the programmer to optimize key sections of their code using assembly language. To achieve these extreme levels of optimization, the DSP engineer must be intimately familiar with the details of their hardware.

As mentioned earlier, many DSP systems include multiple processors. For example, many systems include both a GPP and a DSP. In recent years, multicore processors have also become commonplace. (*Multicore* processors combine two or more processor cores on a single chip.) Thus, multiprocessor programming is a critical skill for many DSP programmers.

Until recently, DSP programmers wrote most of their own software. Today, DSP programmers often use off-the-shelf software for large parts of the system. This is particularly true for speech- and media-processing *codecs*. These codecs have become highly standardized and commoditized.

Obviously, the programmer must understand the algorithms they implement. Programmers who use off-the-shelf DSP software only need to know the basics of the underlying algorithms. In either case, a bachelor's-level education is generally sufficient, but a master's-level education is helpful.

DSP Project Flow

So far we have described the three DSP roles as separate disciplines, but the three roles tend to overlap in practice. For example, a single engineer may fill all three roles in a smaller project. It is also important to note that we have implied a linear project handoff from one discipline to the next. Projects rarely work this way in practice. Instead, most projects follow an iterative process with extensive feedback between the various roles. In addition, the system design, hardware design, and programming often proceed in parallel.

The Future of DSP

DSP was once a narrow, highly specialized field. Five years ago, DSP was nearly synonymous with telecom. If someone told you they were a DSP engineer, you had a good idea of what they did. You could be certain that they had good math skills and could explain exactly how a FIR filter works. Today, DSP is everywhere, often disguised under a moniker like "media processing." Many of the engineers working on DSP systems have only a general understanding of the underlying algorithms. With all these changes, it is hard to clearly define exactly what a DSP engineer is, or what they do—and this confusion is likely to get worse as DSP diffuses into an ever-growing list of applications. However, one thing is certain: DSP engineering will remain an important and in-demand skill for many years to come.

About the Editor

Kenton Williston is the owner of Cabral Consulting. He has been writing about Digital Signal Processing (DSP) technology and business trends since 2001. Kenton is currently the editor of the *DSP DesignLine*, and he was previously the editor and the senior contributor for *Inside DSP*. He is an author of numerous technology reports, including the *Buyer's Guide to DSP Processors*. Kenton is a popular presenter at the Embedded Systems Conference and other venues, and he has advised the leading DSP semiconductor firms on their marketing and product-planning strategies.

In previous lives, Kenton has worked for a major telecommunications company, an engineering services firm, and (believe it or not) an oil company. Kenton received his degree in Electrical Engineering from the University of Missouri-Rolla.

About the Contributors

James Bryant (Chapter 1) is a contributor to *Mixed Signal* and *DSP Design Techniques.*

Rick Gentile (Chapter 9) joined Analog Devices Inc. (ADI) in 2000 as a Senior DSP Applications Engineer. He currently leads the Applications Engineering Group, where he is responsible for applications engineering work on the Blackfin, SHARC and TigerSHARC processors . Prior to joining ADI, Rick was a Member of the Technical Staff at MIT Lincoln Laboratory, where he designed several signal processors used in a wide range of radar sensors. He received a B.S. in 1987 from the University of Massachusetts at Amherst and an M.S. in 1994 from Northeastern University, both in Electrical and Computer Engineering.

Keith Jack (Chapter 5) is Director of Product Marketing at Sigma Designs, a leading supplier of high-performance System-on-Chip (SoC) solutions for the IPTV, Blu-ray, and HDTV markets. Previously, he was Director of Product Marketing at Innovision, focused on solutions for digital televisions. Mr. Jack has also served as Strategic Marketing Manager at Harris Semiconductor and Brooktree Corporation. He has architected and introduced to market over 35 multimedia SoCs for the consumer markets, and is the author of "Video Demystified".

David Katz (Chapter 9) has over 15 years of experience in circuit and system design. Currently, he is Blackfin Applications Manager at Analog Devices, Inc., where he focuses on specifying new processors. He has published over 100 embedded processing articles domestically and internationally, and he has presented several conference papers in the field. Additionally, he is co-author of Embedded Media Processing (Newnes 2005). Previously, he worked at Motorola, Inc., as a senior design engineer in cable modem and automation groups. David holds both a B.S. and M. Eng. in Electrical Engineering from Cornell University.

Nasser Kehtarnavaz (Chapter 3) is the author of *Digital Signal Processing System Design* and *Real-Time Digital Signal Processing.*

Walt Kester (Chapter 1) is a corporate staff applications engineer at Analog Devices. For over 35 years at Analog Devices, he has designed, developed, and given applications support for high-speed ADCs, DACs, SHAs, op amps, and analog multiplexers. Besides writing many papers and articles, he prepared and edited eleven major applications books which form the basis for the Analog Devices world-wide technical seminar series including the topics of op amps, data conversion, power management, sensor signal conditioning, mixed-signal, and practical analog design techniques. He also is the editor of *The Data Conversion Handbook,* a 900+ page comprehensive book on data conversion published in 2005 by Elsevier. Walt has a BSEE from NC State University and MSEE from Duke University.

Dake Liu (Chapter 7) is a Professor and the Director of the Computer Engineering Division in the Department of Electrical Engineering at Linköping University, Linköping, Sweden. His research focuses on architectures of application specific instruction set processors (ASIP) and on-chip multi-core integrations based on VLSI. His research goal is to explore different processor architectures and inter-processor architectures and parallel programming for DSP firmware. On the processor level, his research focus is on integrated DSP-MCU solution based on homogeneous instruction set and heterogeneous architecture. On ILP (Instruction Level Parallelism) level research, his current focus is on architecture and toolchain for template based programming, especially for conflict free parallel memory accesses. On CMP (on Chip Multi Processor) level research, his current focus is on flexible low latency and low silicon cost system for real-time computing. Applications behind his research are embedded DSP computing, communications, and media signal processing. He published about 100 papers on international journals and reviewed conferences.

Robert Meddins (Chapter 2) is the author of *Introduction to Digital Signal Processing.*

Robert Oshana (Chapter 10) has over 25 years of experience in embedded and real-time systems and DSP systems development. He has numerous articles and books in this area. He is a member of the Embedded Systems Advisory board and speaks frequently at Embedded Systems Conferences worldwide. Rob is a senior member of IEEE, a licensed professional engineer (PE), and an adjunct professor at Southern Methodist University.

Ian Poole (Chapter 6) is an established electronics engineering consultant with considerable experience in the communications and cellular markets. He is the author of a number of books on radio and electronics and he has contributed to many magazines in the UK and worldwide. He is also winner of the inaugural Bill Orr Award for technical writing from the ARRL.

Khalid Sayood (Chapter 4) received his BS and MS in Electrical Engineering from the University of Rochester in 1977 and 1979, respectively, and his Ph.D. in Electrical Engineering from Texas A&M University in 1982. In 1982, he joined the University of Nebraska, where he is the Henson Professor of Engineering. His research interests include data compression, joint source channel coding, and bioinformatics.

Dan Sheingold (Chapter 1) is a contributor to *Mixed Signal and DSP Design Techniques.*

Li Tan (Chapter 8) is a Senior Professor and Curriculum Coordinator in Electronics Engineering Technology at DeVry University.

Contents