

Embedded Signal Processing with the Micro Signal Architecture

Woon-Seng Gan
Sen M. Kuo



IEEE PRESS



WILEY-INTERSCIENCE

A John Wiley & Sons, Inc., Publication

**Embedded Signal
Processing with the
Micro Signal Architecture**



THE WILEY BICENTENNIAL—KNOWLEDGE FOR GENERATIONS

Each generation has its unique needs and aspirations. When Charles Wiley first opened his small printing shop in lower Manhattan in 1807, it was a generation of boundless potential searching for an identity. And we were there, helping to define a new American literary tradition. Over half a century later, in the midst of the Second Industrial Revolution, it was a generation focused on building the future. Once again, we were there, supplying the critical scientific, technical, and engineering knowledge that helped frame the world. Throughout the 20th Century, and into the new millennium, nations began to reach out beyond their own borders and a new international community was born. Wiley was there, expanding its operations around the world to enable a global exchange of ideas, opinions, and know-how.

For 200 years, Wiley has been an integral part of each generation's journey, enabling the flow of information and understanding necessary to meet their needs and fulfill their aspirations. Today, bold new technologies are changing the way we live and learn. Wiley will be there, providing you the must-have knowledge you need to imagine new worlds, new possibilities, and new opportunities.

Generations come and go, but you can always count on Wiley to provide you the knowledge you need, when and where you need it!

WILLIAM J. PESCE
PRESIDENT AND CHIEF EXECUTIVE OFFICER

PETER BOOTH WILEY
CHAIRMAN OF THE BOARD

Embedded Signal Processing with the Micro Signal Architecture

Woon-Seng Gan
Sen M. Kuo



IEEE PRESS



WILEY-INTERSCIENCE
A John Wiley & Sons, Inc., Publication

Copyright © 2007 by John Wiley & Sons, Inc. All rights reserved

Published by John Wiley & Sons, Inc., Hoboken, New Jersey
Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4470, or on the web at www.copyright.com. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permission>.

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services or for technical support, please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic formats. For more information about Wiley products, visit our web site at www.wiley.com.

Library of Congress Cataloging-in-Publication Data

Gan, Woon-Seng.

Embedded signal processing with the Micro Signal Architecture / by Woon-Seng Gan
and Sen M. Kuo.

p. cm.

Includes bibliographical references and index.

ISBN: 978-0-471-73841-1

1. Signal processing—Digital techniques.
2. Embedded computer systems—Programming.
3. Computer architecture. I. Kuo, Sen M. (Sen-Maw) II. Title.
TK5102.9.G364 2007
621.382'2 – dc22

2006049693

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Contents

Preface	xi
Acknowledgments	xvii
About the Authors	xix
1. Introduction	1

1.1	Embedded Processor: Micro Signal Architecture	1
1.2	Real-Time Embedded Signal Processing	6
1.3	Introduction to the Integrated Development Environment VisualDSP++	7
1.3.1	Setting Up VisualDSP++	8
1.3.2	Using a Simple Program to Illustrate the Basic Tools	9
1.3.3	Advanced Setup: Using the Blackfin BF533 or BF537 EZ-KIT	12
1.4	More Hands-On Experiments	15
1.5	System-Level Design Using a Graphical Development Environment	18
1.5.1	Setting up LabVIEW and LabVIEW Embedded Module for Blackfin Processors	19
1.6	More Exercise Problems	21

Part A Digital Signal Processing Concepts

2. Time-Domain Signals and Systems	25	
2.1	Introduction	25
2.2	Time-Domain Digital Signals	26
2.2.1	Sinusoidal Signals	26
2.2.2	Random Signals	28
2.3	Introduction to Digital Systems	33
2.3.1	Moving-Average Filters: Structures and Equations	34
2.3.2	Digital Filters	37
2.3.3	Realization of FIR Filters	41

2.4	Nonlinear Filters	45
2.5	More Hands-On Experiments	47
2.6	Implementation of Moving-Average Filters with Blackfin Simulator	50
2.7	Implementation of Moving-Average Filters with BF533/BF537 EZ-KIT	52
2.8	Moving-Average Filter in LabVIEW Embedded Module for Blackfin Processors	54
2.9	More Exercise Problems	57
3. Frequency-Domain Analysis and Processing		59
<hr/>		
3.1	Introduction	59
3.2	The z -Transform	60
3.2.1	Definitions	60
3.2.2	System Concepts	62
3.2.3	Digital Filters	64
3.3	Frequency Analysis	70
3.3.1	Frequency Response	70
3.3.2	Discrete Fourier Transform	76
3.3.3	Fast Fourier Transform	78
3.3.4	Window Functions	83
3.4	More Hands-On Experiments	88
3.4.1	Simple Low-Pass Filters	88
3.4.2	Design and Applications of Notch Filters	91
3.4.3	Design and Applications of Peak Filters	96
3.5	Frequency Analysis with Blackfin Simulator	98
3.6	Frequency Analysis with Blackfin BF533/BF537 EZ-KIT	102
3.7	Frequency Analysis with LabVIEW Embedded Module for Blackfin Processors	105
3.8	More Exercise Problems	110
4. Digital Filtering		112
<hr/>		
4.1	Introduction	112
4.1.1	Ideal Filters	113
4.1.2	Practical Filter Specifications	115
4.2	Finite Impulse Response Filters	120
4.2.1	Characteristics and Implementation of FIR Filters	121
4.2.2	Design of FIR Filters	123
4.2.3	Hands-On Experiments	126
4.3	Infinite Impulse Response Filters	129
4.3.1	Design of IIR Filters	129
4.3.2	Structures and Characteristics of IIR Filters	133
4.3.3	Hands-On Experiments	136

4.4	Adaptive Filters	139
4.4.1	Structures and Algorithms of Adaptive Filters	139
4.4.2	Design and Applications of Adaptive Filters	142
4.4.3	More Hands-On Experiments	148
4.5	Adaptive Line Enhancer Using Blackfin Simulator	151
4.6	Adaptive Line Enhancer Using Blackfin BF533/BF537 EZ-KIT	152
4.7	Adaptive Line Enhancer Using LabVIEW Embedded Module for Blackfin Processors	155
4.8	More Exercise Problems	158

Part B Embedded Signal Processing Systems and Concepts

5. Introduction to the Blackfin Processor 163

5.1	The Blackfin Processor: An Architecture for Embedded Media Processing	163
5.1.1	Introduction to Micro Signal Architecture	163
5.1.2	Overview of the Blackfin Processor	164
5.1.3	Architecture: Hardware Processing Units and Register Files	165
5.1.4	Bus Architecture and Memory	182
5.1.5	Basic Peripherals	187
5.2	Software Tools for the Blackfin Processor	189
5.2.1	Software Development Flow and Tools	189
5.2.2	Assembly Programming in VisualDSP++	191
5.2.3	More Explanation of Linker	195
5.2.4	More Debugging Features	198
5.3	Introduction to the FIR Filter-Based Graphic Equalizer	200
5.4	Design of Graphic Equalizer Using Blackfin Simulator	202
5.5	Implementation of Graphic Equalizer Using BF533/BF537 EZ-KIT	206
5.6	Implementation of Graphic Equalizer Using LabVIEW Embedded Module for Blackfin Processors	211
5.7	More Exercise Problems	214

6. Real-Time DSP Fundamentals and Implementation Considerations 217

6.1	Number Formats Used in the Blackfin Processor	217
6.1.1	Fixed-Point Formats	217
6.1.2	Fixed-Point Extended Format	229
6.1.3	Fixed-Point Data Types	231
6.1.4	Emulation of Floating-Point Format	231
6.1.5	Block Floating-Point Format	235

6.2	Dynamic Range, Precision, and Quantization Errors	236
6.2.1	Incoming Analog Signal and Quantization	236
6.2.2	Dynamic Range, Signal-to-Quantization Noise Ratio, and Precision	238
6.2.3	Sources of Quantization Errors in Digital Systems	240
6.3	Overview of Real-Time Processing	250
6.3.1	Real-Time Versus Offline Processing	250
6.3.2	Sample-by-Sample Processing Mode and Its Real-Time Constraints	251
6.3.3	Block Processing Mode and Its Real-Time Constraints	252
6.3.4	Performance Parameters for Real-Time Implementation	255
6.4	Introduction to the IIR Filter-Based Graphic Equalizer	260
6.5	Design of IIR Filter-Based Graphic Equalizer Using Blackfin Simulator	261
6.6	Design of IIR Filter-Based Graphic Equalizer with BF533/BF537 EZ-KIT	266
6.7	Implementation of IIR Filter-Based Graphic Equalizer with LabVIEW Embedded Module for Blackfin Processors	266
6.8	More Exercise Problems	270

7. Memory System and Data Transfer

274

7.1	Overview of Signal Acquisition and Transfer to Memory	274
7.1.1	Understanding the CODEC	274
7.1.2	Connecting AD1836A to BF533 Processor	276
7.1.3	Understanding the Serial Port	282
7.2	DMA Operations and Programming	287
7.2.1	DMA Transfer Configuration	289
7.2.2	Setting Up the Autobuffer-Mode DMA	291
7.2.3	Memory DMA Transfer	297
7.2.4	Setting Up Memory DMA	298
7.2.5	Examples of Using Memory DMA	298
7.2.6	Advanced Features of DMA	302
7.3	Using Cache in the Blackfin Processor	303
7.3.1	Cache Memory Concepts	303
7.3.2	Terminology in Cache Memory	305
7.3.3	Instruction Cache	307
7.3.4	Data Cache	310
7.3.5	Memory Management Unit	313
7.4	Comparing and Choosing Between Cache and Memory DMA	315
7.5	Scratchpad Memory of Blackfin Processor	317

7.6	Signal Generator Using Blackfin Simulator	317
7.7	Signal Generator Using BF533/BF537 EZ-KIT	319
7.8	Signal Generation with LabVIEW Embedded Module for Blackfin Processors	321
7.9	More Exercise Problems	326

8. Code Optimization and Power Management **330**

8.1	Code Optimization	330
8.2	C Optimization Techniques	331
8.2.1	C Compiler in VisualDSP++	332
8.2.2	C Programming Considerations	333
8.2.3	Using Intrinsics	339
8.2.4	Inlining	342
8.2.5	C/C++ Run Time Library	343
8.2.6	DSP Run Time Library	343
8.2.7	Profile-Guided Optimization	346
8.3	Using Assembly Code for Efficient Programming	349
8.3.1	Using Hardware Loops	352
8.3.2	Using Dual MACs	353
8.3.3	Using Parallel Instructions	353
8.3.4	Special Addressing Modes: Separate Data Sections	355
8.3.5	Using Software Pipelining	356
8.3.6	Summary of Execution Cycle Count and Code Size for FIR Filter Implementation	357
8.4	Power Consumption and Management in the Blackfin Processor	358
8.4.1	Computing System Power in the Blackfin Processor	358
8.4.2	Power Management in the Blackfin Processor	360
8.5	Sample Rate Conversion with Blackfin Simulator	365
8.6	Sample Rate Conversion with BF533/BF537 EZ-KIT	369
8.7	Sample Rate Conversion with LabVIEW Embedded Module for Blackfin Processors	371
8.8	More Exercise Problems	374

Part C Real-World Applications

9. Practical DSP Applications: Audio Coding and Audio Effects **381**

9.1	Overview of Audio Compression	381
9.2	MP3/Ogg Vorbis Audio Encoding	386
9.3	MP3/Ogg Vorbis Audio Decoding	390

9.4	Implementation of Ogg Vorbis Decoder with BF537 EZ-KIT	391
9.5	Audio Effects	393
9.5.1	3D Audio Effects	393
9.5.2	Implementation of 3D Audio Effects with BF533/BF537 EZ-KIT	396
9.5.3	Generating Reverberation Effects	398
9.5.4	Implementation of Reverberation with BF533/BF537 EZ-KIT	399
9.6	Implementation of MDCT with LabVIEW Embedded Module for Blackfin Processors	400
9.7	More Application Projects	404
10. Practical DSP Applications: Digital Image Processing		406
10.1	Overview of Image Representation	406
10.2	Image Processing with BF533/BF537 EZ-KIT	409
10.3	Color Conversion	410
10.4	Color Conversion with BF533/BF537 EZ-KIT	412
10.5	Two-Dimensional Discrete Cosine Transform	413
10.6	Two-Dimensional DCT/IDCT with BF533/BF537 EZ-KIT	416
10.7	Two-Dimensional Filtering	417
10.7.1	2D Filtering	418
10.7.2	2D Filter Design	420
10.8	Two-Dimensional Filtering with BF533/BF537 EZ-KIT	422
10.9	Image Enhancement	422
10.9.1	Gaussian White Noise and Linear Filtering	423
10.9.2	Impulse Noise and Median Filtering	425
10.9.3	Contrast Adjustment	428
10.10	Image Enhancement with BF533/BF537 EZ-KIT	432
10.11	Image Processing with LabVIEW Embedded Module for Blackfin Processors	433
10.12	More Application Projects	438
Appendix A: An Introduction to Graphical Programming with LabVIEW		441
Appendix B: Useful Websites		462
Appendix C: List of Files Used in Hands-On Experiments and Exercises		464
Appendix D: Updates of Experiments Using Visual DSP++ V4.5		473
References		475
Index		479

Preface

In this digital Internet age, information can be received, processed, stored, and transmitted in a fast, reliable, and efficient manner. This advancement is made possible by the latest fast, low-cost, and power-efficient embedded signal processors. Embedded signal processing is widely used in most digital devices and systems and has grown into a “must-have” category in embedded applications. There are many important topics related to embedded signal processing and control, and it is impossible to cover all of these subjects in a one- or two-semester course. However, the Internet is now becoming an effective platform in searching for new information, and this ubiquitous tool is enriching and speeding up the learning process in engineering education. Unfortunately, students have to cope with the problem of information overflow and be wise in extracting the right amount of material at the right time.

This book introduces just-in-time and just-enough information on embedded signal processing using the embedded processors based on the micro signal architecture (MSA). In particular, we examine the MSA-based processors called Blackfin processors from Analog Devices (ADI). We extract relevant and sufficient information from many resources, such as textbooks, electronic books, the ADI website, signal processing-related websites, and many journals and magazine articles related to these topics. The just-in-time organization of these selective topics provides a unique experience in learning digital signal processing (DSP). For example, students no longer need to learn advanced digital filter design theory before embarking on the actual design and implementation of filters for real-world applications. In this book, students learn just enough essential theory and start to use the latest tools to design, simulate, and implement the algorithms for a given application. If they need a more advanced algorithm to solve a more sophisticated problem, they are now more confident and ready to explore new techniques. This exploratory attitude is what we hope students will achieve through this book.

We use assembly programming to introduce the architecture of the embedded processor. This is because assembly code can give a more precise description of the processor’s architecture and provide a better appreciation and control of the hardware. Without this understanding, it is difficult to program and optimize code using embedded signal processors for real-world applications. However, the use of C code as a main program that calls intrinsic and DSP library functions is still the preferred programming style for the Blackfin processor. It is important to think in low-level architecture but write in high-level code (C or graphical data flow). Therefore, we show how to balance high-level and low-level programming and introduce the techniques needed for optimization. In addition, we also introduce a very versatile

graphical tool jointly developed by ADI and National Instruments (NI) that allows users to design, simulate, implement, and verify an embedded system with a high-level graphical data flow approach.

The progressive arrangement makes this book suitable for engineers. They may skip some topics they are already familiar with and focus on the sections they are interested in. The following subsections introduce the essential parts of this book and how these parts are linked together.

PART A: USING SOFTWARE TOOLS TO LEARN DSP—A JUST-IN-TIME AND PROJECT-ORIENTED APPROACH

In Chapters 2, 3, and 4, we explore fundamental DSP concepts using a set of software tools from the MathWorks, ADI, and NI. Rather than introducing all theoretical concepts at the beginning and doing exercises at the end of each chapter, we provide just enough information on the required concepts for solving the given problems and supplement with many quizzes, interactive examples, and hands-on exercises along the way in a just-in-time manner. Students learn the concepts by doing the assignments for better understanding. This approach is especially suitable for studying these subjects at different paces and times, thus making self-learning possible.

In addition to these hands-on exercises, the end of each chapter also provides challenging pen-and-paper and computer problems for homework assignments. These problem sets build upon the previous knowledge learned and extend the thinking to more advanced concepts. These exercises will motivate students in looking for different solutions for a given problem. The goal is to cultivate a learning habit after going through the book.

The theory portion of these chapters may be skipped for those who have taken a fundamental course on DSP. Nonetheless, these examples and hands-on exercises serve as a handy reference on learning important tools available in MATLAB, the integrated development environment VisualDSP++, and the LabVIEW Embedded Module for Blackfin Processors. These tools provide a platform to convert theoretical concepts into software code before learning the Blackfin processor in detail. The introduction to the latest LabVIEW Embedded Module for Blackfin Processors shows the advancement in rapid prototyping and testing of embedded system designs for real-world applications. This new tool provides exciting opportunities for new users to explore embedded signal processing before learning the programming details. Therefore, instructors can make use of these graphical experiments at the end of each chapter to teach embedded signal processing concepts in foundation engineering courses.

PART B: LEARNING REAL-TIME SIGNAL PROCESSING WITH THE BLACKFIN PROCESSOR—A BITE-SIZE APPROACH TO SAMPLING REAL-TIME EXAMPLES AND EXERCISES

Part B consists of Chapters 5, 6, 7, and 8, which concentrate on the design and implementation of embedded systems based on the Blackfin processor. Unlike a conventional user's manual that covers the processor's architecture, instruction set, and peripherals in detail, we introduce just enough relevant materials to get started on Blackfin-based projects. Many hands-on examples and exercises are designed in a step-by-step manner to guide users toward this goal. We take an integrated approach, starting from algorithm design using MATLAB with floating-point simulations to the fixed-point implementation on the Blackfin processor, and interfacing with external peripherals for building a stand-alone or portable device. Along this journey to final realization, many design and development tools are introduced to accomplish different tasks. In addition, we provide many hints and references and supplement with many challenging problems for students to explore more advanced topics and applications.

Part B is in fact bridging the gap from DSP concepts to real-time implementations on embedded processors, and providing a starting point for students to embark on real-time signal processing programming with a fixed-point embedded processor.

PART C: DESIGNING AND IMPLEMENTING REAL-TIME DSP ALGORITHMS AND APPLICATIONS—AN INTEGRATED APPROACH

The final part (Chapters 9 and 10) motivates users to take on more challenging real-time applications in audio signal processing and image processing. Students can use the knowledge and tools learned in the preceding chapters to complete the applications introduced in Chapters 9 and 10. Some guides in the form of basic concepts, block diagrams, sample code, and suggestions are provided to solve these application problems. We use a module approach in Part C to build the embedded system part by part, and also provide many opportunities for users to explore new algorithms and applications that are not covered in Parts A and B. These application examples also serve as good mini-projects for a hands-on design course on embedded signal processing. As in many engineering problems, there are many possible solutions. There are also many opportunities to make mistakes and learn valuable lessons. Users can explore the references and find a possible solution for solving these projects. In other words, we want the users to explore, learn, and have fun!

A summary of these three parts of the book is illustrated in Figure 1. It shows three components: (A) DSP concepts, (B) embedded processor architecture and real-time DSP considerations, and (C) real-life applications: a simple A-B-C

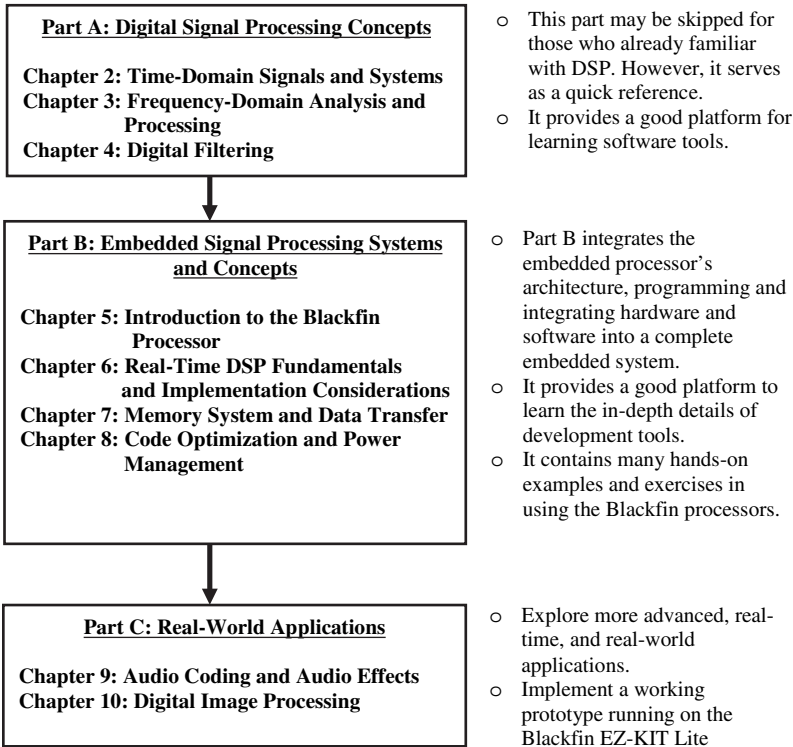


Figure 1 Summary of the book: contents and how to use them

approach to learning embedded signal processing with the micro signal architecture.

DESCRIPTION OF EXAMPLES, EXERCISES, EXPERIMENTS, PROBLEMS, AND APPLICATION PROJECTS

This book provides readers many opportunities to understand and explore the main contents of each chapter through examples, quizzes, exercises, hands-on experiments, exercise problems, and application projects. It also serves as a good hands-on workbook to learn different embedded software tools (MATLAB, VisualDSP++, and LabVIEW Embedded) and solve practical problems. These hands-on sections are classified under different types as follows.

1. **Examples** provide a just-in-time understanding of the concepts learned in the preceding sections. Examples contain working MATLAB problems to illustrate the concepts and how problems can be solved. The naming convention for software example files is


```
example{chapter number}_{example number}.m
```

They are normally found in the directory

```
c:\adsp\chap{x}\MATLAB_ex{x}\
```

where {x} is the chapter number.

2. **Quizzes** contain many short questions to challenge the readers for immediate feedback of understanding.
3. **Experiments** are mostly hands-on exercises to get familiar with the tools and solve more in-depth problems. These experiments usually use MATLAB, VisualDSP++, or LabVIEW Embedded. The naming convention for software experiment files is

```
exp{chapter number}_{example number}
```

These experiment files can be found in the directory

```
c:\adsp\chap{x}\exp{x}_{no.}<option>
```

where {no.} indicates the experiment number and <option> indicates the BF533 or BF537 EZ-KIT.

4. **Exercises** further enhance the student's learning of the topics in the preceding sections, examples, and experiments. They also provide the opportunity to attempt more advanced problems to strengthen understanding.
5. **Exercise Problems** are located at the end of Chapters 1 to 8. These problem sets explore or extend more interesting and challenging problems and experiments.
6. **Application Projects** are provided at the end of Chapters 9 and 10 to serve as mini-projects. Students can work together as a team to solve these application-oriented projects and submit a report that indicates their approaches, algorithms, and simulations, and how they verify the projects with the Blackfin processor.

Most of the exercises and experiments require testing data. We provide two directories that contain audio and image data files. These files are located in the directories `c:\adsp\audio_files` and `c:\adsp\image_files`.

COMPANION WEBSITE

A website, www.ntu.edu.sg/home/ewsgan/esp_book.html, has been set up to support the book. This website contains many supplementary materials and useful reference links for each chapter. We also include a set of lecture slides with all the figures and tables in PowerPoint format. This website will also introduce new hands-on exercises and new design problems related to embedded signal processing. Because the fast-changing world of embedded processors, the software tools and the Blackfin

processor will also undergo many changes as time evolves. The versions of software tools used in this book are:

- MATLAB Version 7.0
- VisualDSP++ Version 4.0
- LabVIEW 8.0
- LabVIEW Embedded Edition 7.1

This website keeps track of the latest changes and new features of these tools. It also reports on any compatibility problems when running existing experiments with the newer version of software.

All the programs mentioned in the exercises and experiments are available for download in the Wiley ftp site: ftp://ftp.wiley.com/public/sci_tech_med/embedded_signal/.

We also include a feedback section to hear your comments and suggestions. Alternatively, readers are encouraged to email us at ewsgan@ntu.edu.sg and kuo@ceet.niu.edu.

Learning Objective:

We learn by example and by direct experience because there are real limits to the adequacy of verbal instruction.

Malcolm Gladwell, *Blink: The Power of Thinking Without Thinking*, 2005

Acknowledgments

We are very grateful to many individuals for their assistance in developing this book. In particular, we are indebted to Todd Borkowski of Analog Devices, who got us started in this book project. Without his constant support and encouragement, we would not have come this far. We would like to thank Erik B. Luther, Jim Cahow, Mark R. Kaschner, and Matt Pollock of National Instruments for contributing to the experiments, examples, and writing in the LabVIEW Embedded Module for Blackfin Processors. Without their strong commitment and support, we would never have been able to complete many exciting demos within such a short time. The first author would like to thank Chandran Nair and Siew-Hoon Lim of National Instruments for providing their technical support and advice. We would also like to thank David Katz and Dan Ledger of Analog Devices for providing useful advice on Blackfin processors. Additionally, many thanks to Mike Eng, Dick Sweeney, Tonny Jiang, Li Chuan, Mimi Pichey, and Dianwei Sun of Analog Devices.

Several individuals at John Wiley also provided great help to make this book a reality. We wish to thank George J. Telecki, Associate Publisher, for his support of this book. Special thanks must go to Rachel Witmer, Editorial Program Coordinator, who promptly answered our questions. We would also like to thank Danielle Lacourciere and Dean Gonzalez at Wiley for the final preparation of this book.

The authors would also like to thank Dahyanto Harliono, who contributed to the majority of the Blackfin hands-on exercises in the entire book. Thanks also go to Furi Karnapi, Wei-Chee Ku, Ee-Leng Tan, Cyril Tan, and Tany Wijaya, who also contributed to some of the hands-on exercises and examples in the book. We would also like to thank Say-Cheng Ong for his help in testing the LabVIEW experiments.

This book is also dedicated to many of our past and present students who have taken our DSP courses and have written M.S. theses and Ph.D. dissertations and completed senior design projects under our guidance at both NTU and NIU. Both institutions have provided us with a stimulating environment for research and teaching, and we appreciate the strong encouragement and support we have received. Finally, we are greatly indebted to our parents and families for their understanding, patience, and encouragement throughout this period.

WOON-SENG GAN AND SEN M. KUO

About the Authors

Woon-Seng Gan is an Associate Professor in the Information Engineering Division in the School of Electrical and Electronic Engineering at the Nanyang Technological University in Singapore. He is the co-author of the textbook *Digital Signal Processors: Architectures, Implementations, and Applications* (Prentice Hall 2005). He has published more than 130 technical papers in peer-reviewed journals and international conferences. His research interests are embedded media processing, embedded systems, low-power-consumption algorithms, and real-time implementations.

Sen M. Kuo is a Professor and Chair at the Department of Electrical Engineering, Northern Illinois University, DeKalb, IL. In 1993, he was with Texas Instruments, Houston, TX. He is the leading author of four books: *Active Noise Control Systems* (Wiley, 1996), *Real-Time Digital Signal Processing* (Wiley, 2001, 2nd Ed. 2006), *Digital Signal Processors* (Prentice Hall, 2005), and *Design of Active Noise Control Systems with the TMS320 Family* (Texas Instruments, 1996). His research focuses on real-time digital signal processing applications, active noise and vibration control, adaptive echo and noise cancellation, digital audio applications, and digital communications.

Note

- A number of illustrations appearing in this book are reproduced from copyright material published by Analog Devices, Inc., with permission of the copyright owner.
- A number of illustrations and text appearing in this book are reprinted with the permission of National Instruments Corp. from copyright material.
- VisualDSP++[®] is the registered trademark of Analog Devices, Inc.
- LabVIEW[™] and National Instruments[™] are trademarks and trade names of National Instruments Corporation.
- MATLAB[®] is the registered trademark of The MathWorks, Inc.
- Thanks to the Hackles for providing audioclips from their singles “Leave It Up to Me” copyright ©2006 www.TheHackles.com

Chapter 1

Introduction

1.1 EMBEDDED PROCESSOR: MICRO SIGNAL ARCHITECTURE

Embedded systems are usually part of larger and complex systems and are usually implemented on dedicated hardware with associated software to form a computational engine that will efficiently perform a specific function. The dedicated hardware (or embedded processor) with the associated software is embedded into many applications. Unlike general-purpose computers, which are designed to perform many general tasks, an embedded system is a specialized computer system that is usually integrated as part of a larger system. For example, a digital still camera takes in an image, and the embedded processor inside the camera compresses the image and stores it in the compact flash. In some medical instrument applications, the embedded processor is programmed to record and process medical data such as pulse rate and blood pressure and uses this information to control a patient support system. In MP3 players, the embedded processor is used to process compressed audio data and decodes them for audio playback. Embedded processors are also used in many consumer appliances, including cell phones, personal digital assistants (PDA), portable gaming devices, digital versatile disc (DVD) players, digital camcorders, fax machines, scanners, and many more.

Among these embedded signal processing-based devices and applications, digital signal processing (DSP) is becoming a key component for handling signals such as speech, audio, image, and video in real time. Therefore, many of the latest hardware-processing units are equipped with embedded processors for real-time signal processing.

The embedded processor must interface with some external hardware such as memory, display, and input/output (I/O) devices such as coder/decoders to handle real-life signals including speech, music, image, and video from the analog world. It also has connections to a power supply (or battery) and interfacing chips for I/O data transfer and communicates or exchanges information with other embedded processors. A typical embedded system with some necessary supporting hardware is shown in Figure 1.1. A single (or multiple) embedded processing core is used to

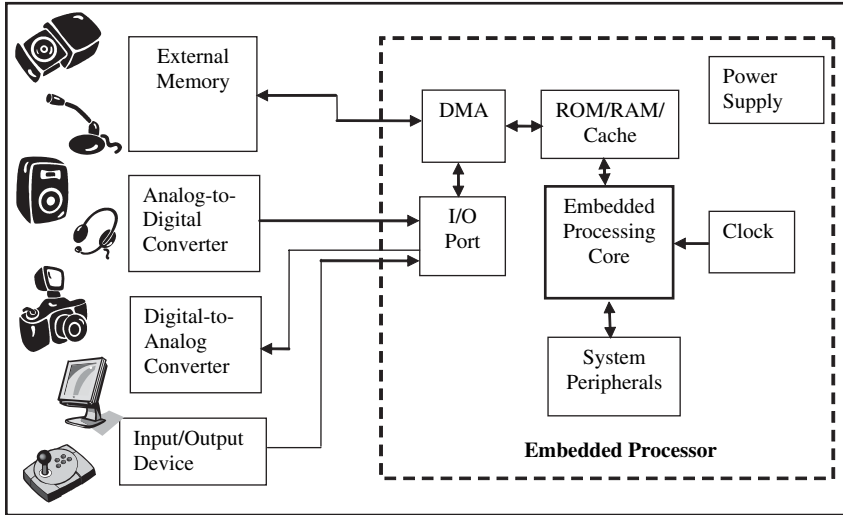


Figure 1.1 Block diagram of a typical embedded system and its peripherals

perform control and signal processing functions. Hardware interfaces to the processing core include (1) internal memories such as read-only memory (ROM) for boot-loading code and random-access memory (RAM) and cache for storing code and data; (2) a direct memory access (DMA) controller that is commonly used to transfer data in and out of the internal memory without the intervention of the main processing core; (3) system peripherals that contain timers, clocks, and power management circuits for controlling the processor's operating conditions; and (4) I/O ports that allow the embedded core to monitor or control some external events and process incoming media streams from external devices. These supporting hardware units and the processing core are the typical building blocks that form an embedded system. The embedded processor is connected to the real-world analog devices as shown in Figure 1.1. In addition, the embedded processor can exchange data with another system or processor by digital I/O channels. In this book, we use hands-on experiments to illustrate how to program various blocks of the embedded system and how to integrate them with the core embedded processor.

In most embedded systems, the embedded processor and its interfaces must operate under real-time constraints, so that incoming signals are required to be processed within a certain time interval. Failure to meet these real-time constraints results in unacceptable outcomes like noisy response in audio and image applications, or even catastrophic consequences in some human-related applications like automobiles, airplanes, and health-monitoring systems. In this book, the terms “embedded processing” and “real-time processing” are often used interchangeably to include both concepts. In general, an embedded system gathers data, processes them, and makes a decision or responds in real time.

To further illustrate how different blocks are linked to the core embedded processor, we use the example of a portable audio player shown in Figure 1.2. In this

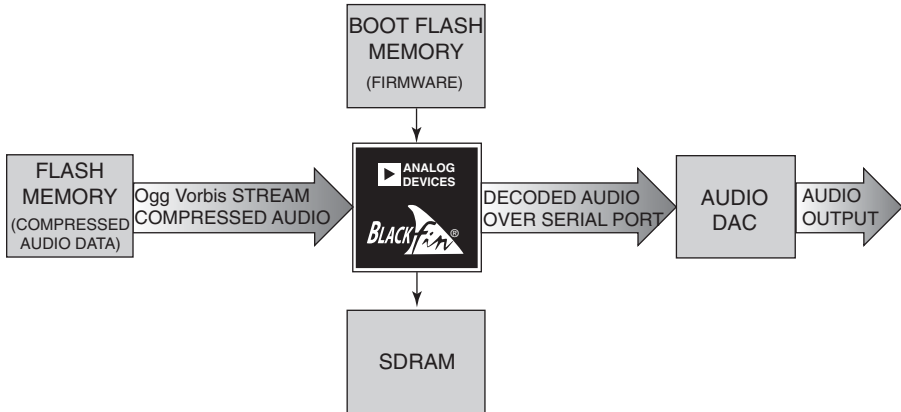


Figure 1.2 A block diagram of the audio media player (courtesy of Analog Devices, Inc.)

system, the compressed audio bit stream in Ogg Vorbis format (introduced in Chapter 9) is stored in the flash memory external to the embedded processor, a Blackfin processor. A decoding program for decoding the audio bit stream is loaded from the boot flash memory into the processor's memory. The compressed data are streamed into the Blackfin processor, which decodes the compressed bit stream into pulse code-modulated (PCM) data. The PCM data can in turn be enhanced by some postprocessing tasks like equalization, reverberation, and three-dimensional audio effects (presented in Chapter 9). The external audio digital-to-analog converter (DAC) converts the PCM data into analog signal for playback with the headphones or loudspeakers.

Using this audio media player as an example, we can identify several common characteristics in typical embedded systems. They are summarized as follows:

1. **Dedicated functions:** An embedded system usually executes a specific task repeatedly. In this example, the embedded processor performs the task of decoding the Ogg Vorbis bit stream and sends the decoded audio samples to the DAC for playback.
2. **Tight constraints:** There are many constraints in designing an embedded system, such as cost, processing speed, size, and power consumption. In this example, the digital media player must be low cost so that it is affordable to most consumers, it must be small enough to fit into the pocket, and the battery life must last for a long time.
3. **Reactive and real-time performance:** Many embedded systems must continuously react to changes of the system's input. For example, in the digital media player, the compressed data bit stream can be decoded in a number of cycles per audio frame (or operating frequency for real-time processing). In addition, the media player also must respond to the change of mode selected by the users during playback.

Therefore, the selection of a suitable embedded processor plays an important role in the embedded system design. A commonly used approach for realizing signal processing tasks is to use fixed-function and hardwired processors. These are implemented as application-specific integrated circuits (ASICs) with DSP capabilities. However, these hardwired processors are very expensive to design and produce, as the development costs become significant for new process lithography. In addition, proliferation and rapid change of new standards for telecommunication, audio, image, and video coding applications makes the hardwired approach no longer the best option.

An alternative is to use programmable processors. This type of processor allows users to write software for the specific applications. The software programming approach has the flexibility of writing different algorithms for different products using the same processor and upgrading the code to meet emerging standards in existing products. Therefore, a product can be pushed to the market in a shorter time frame, and this significantly reduces the development cost compared to the hardwired approach. A programmable digital signal processor is commonly used in many embedded applications. DSP architecture has evolved greatly over the last two decades to include many advanced features like higher clock speed, multiple multipliers and arithmetic units, incorporation of coprocessors for control and communication tasks, and advanced memory configuration. The complexity of today's signal processing applications and the need to upgrade often make a programmable embedded processor a very attractive option. In fact, we are witnessing a market shift toward software-based microarchitectures for many embedded media processing applications.

One of the latest classes of programmable embedded signal processors is the micro signal architecture (MSA). The MSA core [43] was jointly developed by Intel and Analog Devices Inc. (ADI) to meet the computational demands and power constraints of today's embedded audio, video, and communication applications. The MSA incorporates both DSP and microcontroller functionalities in a single core. Both Intel and ADI have further developed processors based on the MSA architecture for different applications. The MSA core combines a highly efficient computational architecture with features usually only available on microcontrollers. These features include optimizations for high-level language programming, memory protection, and byte addressing. Therefore, the MSA has the ability to execute highly complex DSP algorithms and basic control tasks in a single core. This combination avoids the need for a separate DSP processor and microcontroller and thus greatly simplifies both hardware and software design and implementation. In addition, the MSA has a very efficient and dynamic power management feature that is ideal for a variety of battery-powered communication and consumer applications that require high-intensity signal processing on a strict power budget. In fact, the MSA-based processor is a versatile platform for processing video, image, audio, voice, and data.

Inside the computational block of the MSA, there are two multiply-add units, two arithmetic-logic units, and a single shifter. These hardware engines allow the MSA-based processor to efficiently perform several multiply-add operations in

parallel to support complex number crunching tasks. The availability of these hardware units coupled with high clock speed (starts from 300 MHz and rises steadily to the 1-GHz range) has created a substantial speed improvement over conventional microprocessors. The detailed architecture of the MSA core is explained with simple instructions and hands-on experiments in Chapter 5.

The MSA core uses a simple, reduce-instruction-set-computer (RISC)-like instruction set for both control and signal processing applications. The MSA also comes with a set of multifunction instructions that allows different sizes of op-codes to be combined into a single instruction. Therefore, the programmer has the flexibility of reducing the code size, and at the same time, maximizing the usage of available resources. In addition, some special instructions support video and wireless applications. This chapter introduces some basic features of programming and debugging the MSA core and uses examples and exercises to highlight the important features in the software tools. In subsequent chapters, we introduce more advanced features of the software tools.

The MSA core is a fixed-point processor. It operates on fixed-point fractional or integer numbers. In contrast to the floating-point processors, such as the Intel Pentium processors, fixed-point processors require special attention to manipulating numbers to avoid wrong results or extensive computation errors. The concepts of real-time implementation using a fixed-point processor are introduced and examined in Chapter 6.

As explained above, the embedded core must be combined with internal and external memories, serial ports, mixed signal circuits, external memory interfaces, and other peripherals and devices to form an embedded system. Chapter 7 illustrates how to program and configure some of these peripherals to work with the core processor. Chapter 8 explains and demonstrates several important techniques of optimizing the program written for the MSA core. This chapter also illustrates a unique feature of the MSA core to control the clock frequency and supply voltage to the MSA core via software, so as to reduce the power consumption of the core during active operation.

In this book, we examine the MSA core with the latest Blackfin processors (BF5xx series) from ADI. The first generation of Blackfin processors is the BF535 processor, which operates at 300 MHz and comes with L1 and L2 cache memories, system control blocks, basic peripheral blocks, and high-speed I/O. The next generation of Blackfin processors consists of BF531, BF532, BF533, and BF561 processors. These processors operate at a higher clock speed of up to 750 MHz and contain additional blocks like parallel peripheral interface, voltage regulator, external memory interface, and more data and instruction cache. The BF531 and BF532 are the least expensive and operate at 400 MHz, and the BF561 is a dual-core Blackfin processor that is targeted for very high-end applications. The BF533 processor operates at 750 MHz and provides a good platform for media processing applications. Recently released Blackfin processors include BF534, BF536, and BF537. These processors operate at around 400–500 MHz and feature a strong support for Ethernet connection and a wider range of peripherals.

Because all Blackfin processors are code compatible, the programs written for one processor can be easily ported to other processors. This book uses BF533 and BF537 processors to explain architecture, programming, peripheral interface, and implementation issues. The main differences between the BF533 and the BF537 are the additional peripheral supports and slightly less internal instruction memory of the BF537 processors. Therefore, explanation of the Blackfin processing core can be focused solely on the BF533, and additional sections are introduced for the extra peripheral supports in the BF537.

There are several low-cost development tools introduced by ADI. In this book, we use the VisualDSP++ simulator to verify the correctness of the algorithm and the EZ-KIT (development board that contains the MSA processor, memory, and other peripherals) for the Blackfin BF533 and BF537 processors for real-time signal processing and control applications. In addition, we also use a new tool (LabVIEW Embedded Module for Blackfin Processors) jointly developed by National Instruments (NI) and ADI to examine a new approach in programming the Blackfin processor with a blockset programming approach.

1.2 REAL-TIME EMBEDDED SIGNAL PROCESSING

DSP plays an important role in many real-time embedded systems. A real-time embedded system is a system that requires response to external inputs within a specific period. For example, a speech-recognition device sampling speech at 8 kHz (bandwidth of 4 kHz) must respond to the sampled signal within a period of 125 μ s. Therefore, it is very important that we take special attention to define the real-time system and highlight some special design considerations that apply to real-time embedded signal processing systems.

Generally, a real-time system must maintain a timely response to both internal and external signal/data. There are two types of real-time system: hard and soft real-time systems. For the hard real-time system, an absolute deadline for the completion of the overall task is imposed. If the hard deadline is not met, the task has failed. For example, in the case of speech enhancement, the DSP software must be completed within 125 μ s; otherwise, the device will fail to function correctly. In the case of a soft real-time system, the task can be completed in a more relaxed time range. For example, it is not critical how long it takes to complete a credit card transaction. There is no hard deadline by which the transaction must be completed, as long as it is within a reasonable period of time.

In this book, we only examine hard real-time systems because all embedded media processing systems are hard real-time systems. There are many important challenges when designing a hard real-time system. Some of the challenges include:

1. ***Understanding DSP concepts and algorithms.*** A solid understanding of the important DSP principles and algorithms is the key to building a successful real-time system. With this knowledge, designers can program and optimize the algorithm on the processor using the best parameters and set-

tings. Chapters 2 to 4 introduce the fundamentals of DSP and provide many hands-on experiments to implement signal processing algorithms.

2. **Resource availability.** The selection of processor core, peripherals, sensors and actuators, user interface, memory, development, and debugging tools is a complex task. There are many critical considerations that make the decision tough. Chapter 5 shows the architecture of the Blackfin processor and highlights its strength in developing the embedded system.
3. **Arithmetic precision.** In most embedded systems, a fixed-point arithmetic is commonly used because fixed-point processors are cheaper, consume less power, and have higher processing speed as compared to floating-point processors. However, fixed-point processors are more difficult to program and also introduce many design challenges that are discussed in Chapter 6.
4. **Response time requirements.** Designers must consider hardware and software issues. Hardware considerations include processor speed, memory size and its transfer rate, and I/O bandwidth. Software issues include programming language, software techniques, and programming the processor's resources. A good tool can greatly speed up the process of developing and debugging the software and ensure that real-time processing can be achieved. Chapter 7 explains the peripherals and I/O transfer mechanism of the Blackfin processor, whereas Chapter 8 describes the techniques used in optimizing the code in C and assembly programming.
5. **Integration of software and hardware in embedded system.** A final part of this book implements several algorithms for audio and image applications. The Blackfin BF533/BF537 EZ-KITs are used as the platform for integration of software and hardware.


To start the design of the embedded system, we can go through a series of exercises using the development tools. As we progress to subsequent chapters, more design and debugging tools and features of these tools are introduced. This progressive style in learning the tools will not overload the users at the beginning. We use only the right tool with just enough features to solve a given problem.

1.3 INTRODUCTION TO THE INTEGRATED DEVELOPMENT ENVIRONMENT VISUALDSP++

In this section, we examine the software development tool for embedded signal processors. The software tool for the Blackfin processor is the VisualDSP++ [33] supplied by ADI. VisualDSP++ is an integrated development and debugging environment (IDDE) that provides complete graphical control of the edit, build, and debug processes. In this section, we show the detailed steps of loading a project file into the IDDE, building it, and downloading the executable file into the simulator (or the EZ-KIT). We introduce some important features of the VisualDSP++ in this chapter, and more advanced features are introduced in subsequent chapters.

1.3.1 Setting Up VisualDSP++

The Blackfin version of VisualDSP++ IDDE can be downloaded and tested for a period of 90 days from the ADI website. Once it is downloaded and installed, a

VisualDSP++ Environment icon  will appear on the desktop. Double-click on this icon to activate the VisualDSP++. A **New Session** window will appear as shown in Figure 1.3. Select the **Debug target**, **Platform**, and **Processor** as shown in Figure 1.3, and click **OK** to start the VisualDSP++. Under the **Debug target** option, there are two types of Blackfin simulator, a cycle-accurate interpreted simulator and a functional compiled simulator. When **ADSP_BF5xx Blackfin Family Simulators** is selected, the cycle-accurate simulator is used. This simulator provides a more accurate performance, and thus is usually used in this book. The compiled simulator is used when the simulator needs to process a large data file. The **Processor** option allows users to select the type of processor. In this book, only the Blackfin BF533 and BF537 processors are covered. However, the code developed for any Blackfin processor is compatible with other Blackfin processors. In Figure 1.3, the ADSP-BF533 simulator is selected. Alternatively, users can select the ADSP-BF537 simulator.

A VisualDSP++ Version 4 window is shown in Figure 1.4. There are three subwindows and one toolbar menu in the VisualDSP++ window. A **Project Window** displays the files available in the project or project group. The **Disassembly** window displays the assembly code of the program after the project has been built. The **Output Window** consists of two pages, **Console** and **Build**. The **Console** page displays any message that is being programmed in the code, and the **Build** page shows any errors encountered during the build process. The toolbar menu contains all the tools, options, and modes available in the VisualDSP++ environment. We will illustrate these tools as we go through the hands-on examples and exercises in

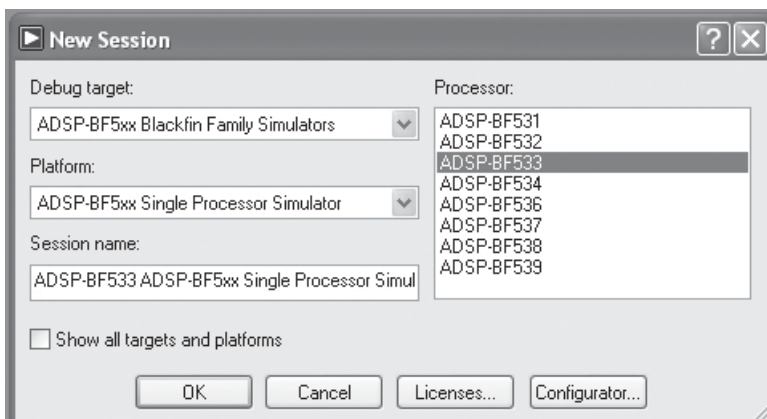


Figure 1.3 A **New Session** window