

Studies in Fuzziness and Soft Computing

Volume 334

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

About this Series

The series “Studies in Fuzziness and Soft Computing” contains publications on various topics in the area of soft computing, which include fuzzy sets, rough sets, neural networks, evolutionary computation, probabilistic and evidential reasoning, multi-valued logic, and related fields. The publications within “Studies in Fuzziness and Soft Computing” are primarily monographs and edited volumes. They cover significant recent developments in the field, both of a foundational and applicable character. An important feature of the series is its short publication time and world-wide distribution. This permits a rapid and broad dissemination of research results.

More information about this series at <http://www.springer.com/series/2941>

Pedro Ponce-Cruz · Arturo Molina
Brian MacCleery

Fuzzy Logic Type 1 and Type 2 Based on LabVIEW™ FPGA

 Springer

Pedro Ponce-Cruz
Tecnologico de Monterrey
Campus Ciudad de México
Tlalpan, Distrito Federal
Mexico

Brian MacCleery
National Instruments Corporation
Austin, TX
USA

Arturo Molina
Tecnologico de Monterrey
Campus Ciudad de México
Tlalpan, Distrito Federal
Mexico

The Fuzzy Logic Type 1 and Type 2 Based on LabVIEW FPGA Toolkit can be downloaded from the additional material of the book. For decompressing the toolkit you have to use the password (TOOLKITFPGA@TEC01). On the other hand, it is allowed to use the toolkit in academic and research implementations based on LabVIEW FPGAs but it has to be properly referenced. It is not allowed to use it in industrial applications without permission from the authors.

ISSN 1434-9922 ISSN 1860-0808 (electronic)
Studies in Fuzziness and Soft Computing
ISBN 978-3-319-26655-8 ISBN 978-3-319-26656-5 (eBook)
DOI 10.1007/978-3-319-26656-5

Library of Congress Control Number: 2015955366

Springer Cham Heidelberg New York Dordrecht London
© Springer International Publishing Switzerland 2016

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

LabVIEW™ is a trademark of National Instruments Corporation, 11500 N Mopac Expwy, Austin, TX 78759-3504, USA, <http://www.ni.com/>.

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

*To Norma, Jamie, Pedro, Alize, Jorge, Aura,
and Giovanna. Also, I dedicate this book
to my lovely mother (Margarita) and
grandmother (Catalina Vazquez[†]) who help
me to dream in fuzzy colors*

Pedro Ponce-Cruz

*To my father Arturo, in memory of my mother
Rosita and my family, Silvia my lovely wife
and beloved kids Julio and Monse*

Arturo Molina

*For Eva Jane MacCleery. Most things in life
and science have a fuzzy type of logic.
However, if we listen carefully and are very
fortunate, we find love, a mysterious clarity to
guide us through this fuzzy world*

Brian MacCleery

Foreword

Fuzzy logic is widely applied in every aspect of our daily lives. Washing machines, air conditioners, and vehicles are examples of applications in which fuzzy controllers are embedded to achieve smooth, nonlinear, and robust control. Such applications depend on a system architecture that is easy to implement, rather than on a theoretically abstract and complex methodology. Therefore, several tools, such as the NI LabVIEW PID and Fuzzy Logic Toolkit for Windows, have been proposed to assist developers in implementing fuzzy logic systems. Some tools for this purpose can even be downloaded for free and come with a user manual. Through the use of such tools, several successful applications have been demonstrated, which has further motivated the application of fuzzy logic in practice. However, most existing fuzzy logic applications in practice use ordinal fuzzy sets. Other fuzzy set types, such as interval-valued fuzzy sets, type-2 fuzzy sets, and hesitant fuzzy sets, have seldom been used. This is mainly because of the difficulties associated with implementing these generalized forms of fuzzy sets. However, several attempts have indicated that using these generalized fuzzy sets has several advantages. For example, the uncertainty behind a phenomenon can be more effectively described using a type-2 fuzzy set.

The authors of this book combined the latest research findings and practical experience with LabVIEW™ FPGA. I particularly appreciate the perspective on FPGA's most recent development strategy. This book also provides information for engineers seeking to understand fuzzy logic and how it can be applied to their products or system designs.

Prof. Dr. Tin-Chih Toly Chen
Outstanding Professor
Feng Chia University
Founding Editor-in-Chief
International Journal of Fuzzy System Applications

Preface

This book presents fuzzy logic and LabVIEW FPGA for designing fuzzy logic controllers. This is a book for implementing fuzzy logic controllers in LabVIEW FPGAs.

Despite the FPGA's attractive features, their adoption by industrial control and signal processing engineers has been slower than processors and DSPs. This is due to several factors. First, these engineers traditionally programmed processors and DSPs using higher level languages, such as C. However, FPGAs possessed complex development tool chains that required designs to be specified using hardware description level (HDL) and register transfer level (RTL) semantics. Furthermore, traditional FPGA development tools lacked intellectual property (IP) blocks for common industrial applications, such as ADC and encoder interface logic, PWM and commutation logic, timing and triggering functions, PID control algorithms, memory management, and data transfer functions. In addition, FPGAs natively supported integer data types only which significantly increased development complexity for analog control and signal processing applications that required math, control, and digital signal processing algorithms, as opposed to floating point processors. Also, traditional FPGA simulation tools were operated at the digital design level and were not interoperable with the type of dynamic simulation tools used by control systems and signal processing engineers for modeling continuous time dynamic system response. Moreover, FPGAs compilation times were relatively long, as compared to processors and DSPs. For example, typical FPGA compilation times today range from 15 to 90 min, whereas processor and DSP compilations are typically completed in less than one minute. Finally, the sequential text-based semantics of traditional register level development tools made it relatively difficult to specify timing and concurrency among parallel processing tasks in a way that leverages the inherent parallel processing capability of FPGA devices.

Despite these traditional development challenges, the successful adoption of FPGAs in application areas such as consumer electronics, and the resulting drop in the price of FPGAs has spurred the interest of industrial control design and simulation vendors. Such vendors are creating the next generation FPGA development

tools that are designed for engineers with little or no digital design expertise. The goal of these next generation “system-level” graphical design tools is to empower control, simulation, and signal processing engineers to harness the full power of the FPGA technology. Graphical system design tools are intended to provide a more intuitive, high level programming paradigm that simplifies the creation of complex parallel processing and control applications. Also, they are intended to provide relatively competitive performance and resource usage, as compared to traditional HDL development tools.

Graphical data flow programming languages are a natural fit for FPGA development due to their inherent sense of parallelism and concurrency that naturally maps to hardware design. Also, recent technological advances are enabling designers to place their FPGA code within a high-level dynamic simulation environment. This ability to cross the boundary between the digital domain of the FPGA and the analog multi-physics domain of the system is facilitating a “true” mechatronics approach to development, in which the complex interplay between FPGA silicon logic, power electronic components, electric motor drives, and mechanical systems can all be simulated in a virtual environment without the need to wait for long FPGA compilations.

The ability to quickly iterate and optimize the FPGA logic design in a mechatronics simulation environment, combined with the new high-level programming tools for FPGAs is reducing dramatically the barriers that prohibited wide adoption of FPGAs in industrial control.

In addition to the improved design and simulation tools for FPGAs, the next generation tools are providing a rapidly growing library of IP blocks for common control and DSP algorithms through code sharing services. On the other hand, the number of books that present Fuzzy logic Control is big as Fuzzy logic control is one of the most important control techniques. However, several books are only mathematical descriptions and are not focused on implementation of fuzzy logic control. Moreover, there are not enough books that deal with implementing fuzzy logic controllers in FPGAs. There is still a lot of vagueness and misunderstanding around the implementation of fuzzy logic controllers implemented in LabVIEW™ FPGA.

Since this book presents a clear description of fuzzy logic control type 1 and 2 that are the most used fuzzy logic representations, the implementation in LabVIEW™ FPGA can be developed. Several experimental examples are presented in order to show the potential of Fuzzy Logic controllers implemented in FPGA.

Finally, a complete LabVIEW™ FPGA toolkit for fuzzy logic type 1 and type 2 is included in the book. This toolkit is based on fix point representation that LabVIEW™ FPGA needs. This toolkit is developed for working on LabVIEW™ real-time systems.

Pedro Ponce-Cruz
Arturo Molina
Brian MacCleery

Acknowledgments

We hereby acknowledge the following organizations for their contributions to this book: Tecnológico de Monterrey, Campus Ciudad de México and National Instruments, Austin, Texas.

Contents

1 Literature Review for Digital Implementations of Fuzzy Logic Type-1 and Type-2.	1
1.1 Advances in Applications of Fuzzy Logic Systems.	1
1.2 FPGA and Microcontrollers Used for Fuzzy Logic Applications.	6
1.2.1 Microcontroller Application.	7
1.2.2 DSP Application	7
1.2.3 FPGA Application	9
1.3 Fuzzy Logic Concepts	10
1.3.1 Type-1 Fuzzy Set (T1Fs)	13
1.3.2 Membership Function.	14
1.3.3 Discourse Universe and Membership Degree.	20
1.4 Extension Principle	20
1.4.1 Basic Identities	22
1.5 Fuzzy Logic Rules.	22
1.6 Defuzzification Methods	23
1.7 Fuzzy Inference Methods	25
1.8 Takagi-Sugeno-Kang	29
1.9 Numerical Example (Mandani)	32
1.10 Basic Numerical Example (TSK)	35
1.11 Type-2 Fuzzy Logic Set	36
1.11.1 Historical Review of Advances	36
1.11.2 Type-2 Fuzzy Sets (T2FS)	37
1.11.3 Footprint of Uncertainty	39
1.12 Fuzzy Sets Type 2 Representations	39
1.12.1 Digital and Continuous Representation	39
1.13 Interval Type 2 Fuzzy Sets (IT2FS)	42
1.14 Type Reduction and Defuzzification.	44
1.14.1 Karnik–Mendel Iterative Procedure (KM)	44
1.14.2 Wu-Mendel Uncertain Bounds.	46
1.14.3 Enhanced Karnik–Mendel Algorithm	47

- 1.14.4 Type 2 Fuzzy Logic Systems Block Diagram 49
- 1.14.5 Interval Type 2 Fuzzy Logic Numeric Example 50
- 1.15 Experimental Implementation of a Fuzzy Logic Controller
 - Type-2 in Quadrotors 54
 - 1.15.1 Introduction. 54
 - 1.15.2 Quadrotor Basic Principles 55
 - 1.15.3 ANFIS 56
- 1.16 Design of Fuzzy Logic Controller Tuned by an Expert 57
- 1.17 Design of Fuzzy Logic Controller Tuned by an Anfis 62
- 1.18 Experimental Results 64
- References 67
- 2 LabVIEW™ FPGA 71**
 - 2.1 Field-Programmable Gate Array (FPGA). 71
 - 2.1.1 How Do FPGA-Based Control Systems Compare to Processor-Based Systems? 72
 - 2.1.2 How Do I Program My Control Application Using the LabVIEW FPGA Module? 74
 - 2.1.3 How Does the LabVIEW Compiler Translate My Graphical Code into FPGA Circuitry? 76
 - 2.1.4 FPGAs Are Fast, but How Do Faster Loop Rates Improve Control System Performance? 77
 - 2.1.5 What FPGA Hardware Targets Are Available from NI? 78
 - 2.1.6 What Closed-Loop Control Performance Can I Achieve? 80
 - 2.1.7 How Much Jitter Can I Expect in My FPGA-Based Control Loops? 81
 - 2.1.8 Creating a New LabVIEW Real-Time Project and Adding I/O 82
 - 2.2 Developing the LabVIEW FPGA Application 90
 - 2.3 Compiling the FPGA Application 101
 - 2.3.1 Understanding the LabVIEW FPGA Compilation Process 102
 - 2.3.2 FPGA Clock Speed 103
 - 2.3.3 The Compilation Report 103
 - 2.4 Advanced Methods for LABVIEW FPGA. 104
 - 2.4.1 Introduction. 105
 - 2.4.2 Technique 1: Use Single-Cycle Timed Loops (SCTLs) 106
 - 2.4.3 Creating Counters and Timers 110
 - 2.4.4 Write Your FPGA Code as Modular, Reusable SubVis 111
 - 2.4.5 Separate Logic from I/O 114
 - 2.4.6 Holding State Values in a Function Block. 115

- 2.4.7 Run-Time Updateable Look-up Table (LUT) 117
- 2.4.8 Do not Place Delay Timers in the SubVI 119
- 2.4.9 Reentrancy 120
- 2.5 Use Simulation Before You Compile 122
 - 2.5.1 Providing Tick Count Values for Simulation 123
 - 2.5.2 Test the LabVIEW FPGA Code Using the LabVIEW Control Design & Simulation Module. 125
- 2.6 Synchronize Your Loops. 128
 - 2.6.1 Latching Values. 129
 - 2.6.2 Application Example 130
- 2.7 Technique 5: Avoid “Gate Hogs”. 132
 - 2.7.1 Avoid Front Panel Arrays for Data Transfer 133
 - 2.7.2 Use DMA for Data Transfer 134
 - 2.7.3 Use the Minimum Data Type Necessary 135
 - 2.7.4 Optimizing for Size 135
 - 2.7.5 Additional Techniques to Optimize Your FPGA Applications 138
- References 138
- 3 Real-Time Fuzzy Logic Controllers. 139**
 - 3.1 Basic Parts in Real-Time Fuzzy Logic Controllers 139
 - 3.2 Case Study: The Karnik–Mendel Algorithms Performance Implemented in Real-Time LABVIEW FPGA 140
 - 3.2.1 Interval Type-2 Fuzzy Logic Systems. 141
 - 3.2.2 The Karnik–Mendel Algorithm 142
 - 3.2.3 Non-iterative Version 142
 - 3.2.4 Iterative Version 144
 - 3.2.5 Enhanced Karnik–Mendel Algorithm 146
 - 3.2.6 Nie-Tan Method 147
 - 3.3 DC Servomotor 148
 - 3.3.1 Laplace Transform Model 149
 - 3.3.2 State-Space Transfer Function 150
 - 3.3.3 Servomotor Control System. 151
 - 3.4 The Hardware Complexity 152
 - 3.5 Methodology. 153
 - 3.6 Results and Discussion 155
 - 3.6.1 Reference Tracking 155
 - 3.6.2 The Hardware Performance 155
 - References 158
- 4 Fuzzy Logic Type 1 and Type 2 LabVIEW FPGA Toolkit. 159**
 - 4.1 Type-1 Fuzzy Sets 159
 - 4.1.1 Membership Function Parameters. 160
 - 4.1.2 Normalization 161

4.1.3	Membership Degree	161
4.1.4	Error Handling.	161
4.2	Type-2 Fuzzy Sets	162
4.2.1	Membership Function Parameters.	162
4.2.2	Normalization	162
4.2.3	Uncertainty Widths	163
4.2.4	Membership Degrees	163
4.2.5	Error Handling.	164
4.2.6	Examples	166
4.3	Creating a Knowledge Base	169
4.3.1	Building a Rule Set	169
4.4	The Inferred Set.	171
4.5	Defuzzification	180
4.5.1	T1 Mamdani Model the Centroid	180
4.5.2	T2 Mamdani Model the Karnik–Mendel Algorithm	181
4.5.3	The Enhanced Karnik–Mendel Algorithm	182
4.5.4	The Nie–Tan Method	182
4.5.5	The Takagi–Sugeno Model	183
4.6	Examples	186
4.7	Study Cases	187
4.7.1	T1FLS Validation	187
4.7.2	Electric Wheelchair	192
4.8	T2FLS Validation	200
4.9	Performance T1 FLS DC Servomotor.	202
4.9.1	Electric Wheelchair	203
4.10	T1FLS Versus T2FLS	204
4.10.1	Noise Response	204
4.10.2	Response Time	206
4.10.3	Resource Utilization	206
4.11	Included Examples.	210
4.11.1	Case Study: Experimental CNC Micromachine Controlled by Fuzzy Type 2	210
4.11.2	Micromachines and Fuzzy Logic	212
4.11.3	Reconfigurable Micromachine Tools.	213
4.11.4	Motion Control	215
4.11.5	Control Design on Real-Time FPGA	217
4.11.6	Experimental Results	222
	References	228
	Index	231