Model-Driven Design Using IEC 61499

Li Hsien Yoong • Partha S. Roop • Zeeshan E. Bhatti
Matthew M.Y. Kuo

Model-Driven Design Using IEC 61499

A Synchronous Approach for Embedded and Automation Systems

Springer

Li Hsien Yoong
Invenco Group Ltd.
Auckland, New Zealand

Zeeshan E. Bhatti
Department of Electrical and Computer Engineering
The University of Auckland
Auckland, New Zealand

Partha S. Roop
Department of Electrical and Computer Engineering
The University of Auckland
Auckland, New Zealand

Matthew M.Y. Kuo
Department of Electrical and Computer Engineering
The University of Auckland
Auckland, New Zealand

ISBN 978-3-319-10520-8 ISBN 978-3-319-10521-5 (eBook)
DOI 10.1007/978-3-319-10521-5
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014952451

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

– Li Hsien: To Sharain, my soulmate, whose unwavering support has been my constant encouragement.

– Partha: To Mother Saraswati and her loving manifestations who have filled this life with love, especially Laxmi, Bindu and Adya and all my family members.

– Zeeshan: To my father Muhammad Ejaz who helped me explore the limits of my abilities, and continues to do so till this day.

– Matthew: To my family Bill, Angela and Vicki for always supporting me.

Foreword

This book summarizes almost a decade of work by Dr. Partha Roop and his talented students on giving a synchronous semantics to block diagrams of the IEC 61499 standard. I was an observer and even a participant of this work. Actually, I think I can take the pride of introducing the concept of IEC 61499 to Partha around 2005 and I am very glad to see how far this got.

The block diagram way of thinking, which the standard explores, has intuitive attraction to many control and automation engineers. The standard extends the block diagram way of systems design to the thinking about distributed systems. The search for better design models has been going on for quite a while in the embedded systems domain as well as in the industrial automation domain especially to design control systems. Despite many similarities, so far these domains have been quite isolated from each other. Embedded systems researchers explored many model-driven engineering concepts, possibly stemming from UML (e.g., SCADE and SIMULINK); however, it appears that these attempts are isolated and do not inherit legacy of progressive evolution. It was, therefore, my impression from the decade back that the concept of event-driven block diagrams from automation systems can give embedded systems researchers a large body of knowledge to further explore. Therefore, this book is a great evidence of convergence of embedded systems and automation systems. What is especially interesting is that it demonstrated influence of automation systems legacy on the embedded systems research. In most cases it has been the other way around.

In my view, the combination of block diagrams and state machines in IEC 61499 presents a good equilibrium of model richness and simplicity. In principle, this language presents a fully executable model. One major limitation has been, however, that the standard did not define execution semantics rigorously, leaving room for interpretations. This open backdoor allowed Partha to investigate applicability of the Esterel's synchronous model to IEC 61499. It was demonstrated by Partha that synchronous execution has excellent performance characteristics.

I am sure the reader will benefit from this book in many ways. It can be used as a problem-based introduction to IEC 61499, with extensions to model-driven design, formal verification, and static timing analysis. The nice results presented in the book will open horizons for many future challenges of dependable distributed systems design.

Aalto University, Espoo, Finland
Luleå University, Luleå, Sweden
Valeriy Vyatkin

Preface

Industrial control systems have evolved over decades to address the ever-increasing complexity and related technical challenges. As the complexity and capabilities of industrial systems grew, racks of relays used to implement ladder logic became too cumbersome for rewiring and troubleshooting. *Programmable logic controllers* (PLCs) replaced this technology, which were easy to configure and debug, specifically where high-speed Boolean control was required. Growing size and scale of industrial equipment created the need for remote I/O (e.g., SCADA) and distributed control systems (e.g., supervisory level DCS), thus making networking technology an integral part of automation systems. In contrast to automation systems, *embedded systems* were typically used to handle application-specific control requirements and were wholly encapsulated by the *equipment under control* (EUC). Low-power, 8-bit processors with built-in peripherals, called *microcontrollers*, were traditionally used for low-demand systems where system availability was not critical. However in the recent years, both the reliability and capabilities of embedded systems have significantly increased. Decreasing cost of embedded equipment and their increasing capabilities such as on-board support for various types of networks have blurred the line between *industrial control and embedded control*.

In many aspects, the two genres of systems are similar, i.e., both use *programmable electronics* [62] to control adjoining equipment referred to as the environment and the EUC. In spite of this similarity, these two domains have remained far apart due to practical reasons. For example, PLCs are the preferred platforms for distributed control not only due to their reliability and remote I/O but also for their excellent tolerance against noise and wide operating temperature ranges. In addition, PLCs offer the capability for interfacing with specialized modules such as electrical drives using fieldbuses. Apart from the differences in applications and interfaces, the two genres of systems also vary in their respective approaches for programmability and execution. Since many industrial applications are real time in nature, the *time-triggered* execution mechanism of PLCs, called the *PLC scan cycle*, is deemed more appropriate. As opposed to this the *event-triggered* approach based on interrupts is widely used in embedded systems. The programmable aspect of control systems implies that the control logic of the system is implemented as a software using an appropriate programming language and design paradigm. PLCs traditionally used simplistic low-level programming languages such as *ladder logic* and *structured text*. However, in recent years, languages inspired by the object-oriented design have also been introduced. Despite this fact, many PLCs are still programmed using older methods.

Embedded systems traditionally relied on 8 and 16-bit microcontrollers for developing applications, mainly for consumer electronics. The main programming language for such systems has been the C language, which is similar in many respects to PLC programming languages (when comparing their level of abstraction). C has been known as an *intermediate* language as it offers many features of low-level assembly languages combined with constructs of high-level programming. Such features are very suitable while designing small- to medium-scale embedded systems, where the software design has to be heavily influenced by the underlying hardware architecture.

The landscape across both domains is rapidly changing. With the widespread usage of cyber-physical systems (CPS) [81] in aviation, automotive, medical devices, process control, and robotics, the complexity of control systems has grown immensely. An even greater challenge is that such systems must ensure *functional safety* [62], i.e., they must operate within established ranges of acceptable risks for the entire lifetime of the system. Existing design approaches based on using low-level languages are not amenable to the design of CPS. The use of such languages and associated design practices increases the possibility of catastrophic events. For example, in the domain of *implantable medical devices*, there were more than 5,000 recalls and more than one million adverse events from 2006 to 2011 as reported by an FDA database. Coincidentally, *over 90 % of these recalls were related to software failures*.

The motivation behind compiling this monograph is linked to the momentum of change in *programmable electronics* in recent years. System on programmable chips (SOPCs) consisting of multi-core processors and field-programmable gate arrays (FPGA) are part of the same fabric. Similarly, programmable automation controllers (PACs) are also rapidly changing the automation landscape, especially with the boom in Internet of Things (IoT). Ever more capable and complex embedded systems are being deployed to perform control operations, and the potential for using embedded solutions for the automation domain is unlimited.

The rapid convergence of embedded systems and automation systems requires reliable and efficient design approaches. *Model-driven approaches* provide assurance for these requirements by using visual languages that are built on mathematical principles. Systems are designed as models of functionality (control) that are combined with the models of environment (plant model). Mathematically sound foundation and modular design thus offer promise for scalable, maintainable, reusable, observable, and testable designs. Examples include Simulink/Stateflow, SCADE, and LabView. Similar approaches based on automation standards are also needed. While visual, software engineering-inspired practices have become common place in the embedded domain, the automation domain has been somewhat slow to embrace the changes happening in the world of software engineering.

A notable exception, however, has been the proposal by the International Electrotechnical Commission (IEC) of the IEC 61499 standard in 2005. This standard bears many similarities with component-oriented software engineering standards such as UML, while also supporting legacy platforms and code based on PLCs. We believe that IEC 61499 is a key enabler for facilitating the said convergence between automation systems and embedded systems. *A model-driven approach using this standard, which facilitates platform and network agnostic designs, can be a game changer for both genres of systems*. This is the key theme of this monograph.

The approach expounded in this monograph is based on our research findings and associated tools using the synchronous approach for IEC 61499. This approach was developed for *safety-critical* embedded systems in the 1980s and led to the proposal of three synchronous languages called *Esterel*, *Lustre*, and *Signal*. Subsequently, Esterel Technologies (www.esterel-technologies.com) commercialized Lustre in the form of an equivalent visual language based on block diagrams called SCADE. This toolchain could perform automatic code generation from SCADE models so as to be compliant with the DO-178B standard. Hence, SCADE is widely used for aircraft flight control (e.g., Airbus A380), rail transportation, and many other domains.

Coincidentally, the synchronous approach uses a time-triggered way of execution very similar to the PLC scan cycle. Hence, we adopted this approach in our early proposal [145] for the execution of function blocks. We developed an operational semantics of function blocks that ensured determinism and deadlock freedom [147]. Subsequently, we developed compilers from function blocks to various languages such as Esterel, Java, C, and PLC languages. These compilers used the developed semantics as the basis of code generation. Because of the synchronous approach, the generated code is not reliant on any run-time system for event scheduling. We have also developed a number of static analysis techniques to further enhance the proposed approach for CPS. More recently, we have also developed an approach for functional safety analysis of automation systems using IEC 61499 [22].

The proposed approach, unlike existing approaches, can generate code on any platform that supports a C compiler. We can also generate PLC code [111]. Our designs have been tried on many PACs. We have tried several embedded processors such as ARM, PIC, Atmel, and soft-core processors such as NiOS, MicroBlaze, and many others. Also, we support a code generator that uses abstract communication patterns and can support any networking protocol. Our approach is especially suited for the design of CPS [148] due to the nature of the code generators and associated static analysis techniques.

This monograph brings together our research findings and associated tools for the readers from both domains. Automation engineers, who are already familiar with earlier generation function blocks (defined in IEC 61131-3), can rapidly adopt the new approach based on familiar syntax and PLC-like execution semantics. Embedded systems designers, familiar with C and block diagram languages such as UML, SCADE, and Simulink, can seamlessly adopt IEC 16499 specifications. We provide a set of pedagogic examples from both domains as case studies and provide hands-on tutorials on our companion site (www.timeme.io). For interested academics, who want to use this as a text in courses on distributed systems, automation systems, or embedded systems, we offer all these resources and will offer lecture slides in the near future.

Readers of the first edition of this book may discover some errors and omissions. Your input to improve our efforts will be highly appreciated and we hope that you enjoy this offering.

Auckland, New Zealand
August 2014

Li Hsien Yoong
Partha S. Roop
Zeeshan E. Bhatti
Matthew M.Y. Kuo

Acknowledgements

The authors would like to acknowledge the following individuals, who have been either our collaborators, coauthors, or people with whom we have discussed regarding the proposed research:

- Professor Valeriy Vyatkin, Aalto University and Luleå University. He was instrumental in introducing the authors to the IEC 61499 standard and also cooperated with us on many interesting research problems.
- Dr. Alois Zoitl, Fortiss, Munich. Partha has interacted with Alois over many years, and he has been very kind to give some feedback on the current manuscript in spite of his busy schedule.
- Dr. Gareth D Shaw, Navman, New Zealand. Gareth has been part of our research on the standard and has proposed the idea of hierarchical and concurrent ECCs (HCECCs), which is part of our tools. Gareth's work, though not part of this book, is a core part of the synchronous approach for function blocks developed by us.
- Mr. Yu Zhao (aka Tony), who developed a model-driven approach for the design of implantable medical devices using IEC 61499. We thankfully acknowledge that the pacemaker case study is based on Tony's work in his Master of Engineering thesis.
- Professor Zoran Salcic, University of Auckland, New Zealand. Zoran has co-supervised Li Hsien Yoong's thesis and has been a coauthor on several papers.
- Dr. Roopak Sinha, AUT University, Auckland, New Zealand. Roopak also cooperated with Gareth and Zeeshan on formal semantics of function blocks based on the synchronous approach.
- Mr. Jiajie Wang (aka Hugh), who is working on timing analysis of function blocks and other synchronous languages at the University of Auckland as a PhD student. He has developed some nice tutorials for our tools, which are available on our companion site (www.timeme.io).
- Mr. Eugene Yip, who is a PhD student at The University of Auckland working on mixed-criticality systems. Eugene gave many interesting feedback on the chapters. He also conceptualized a key figure in the introduction to compare embedded systems with automation systems.

Contents