TOWARDS VERIFIED SYSTEMS

EDITED BY
Jonathan Bowen

TOWARDS VERIFIED SYSTEMS

EDITED BY
Jonathan Bowen

safemos

This page deliberately left blank for publisher's use

This page deliberately left blank for publisher's use

This page deliberately left blank for publisher's use

This page deliberately left blank for publisher's use

Contents

List of Figures

List of Tables

To err is human but to really foul things up requires a computer.

Farmers' Almanac for 1978, *Capsules of Wisdom (1977)*

Foreword

In basic science, fundamental discoveries are made by intense concentration on a single issue, and by rigorous control of all extraneous variation. By contrast, in practical engineering new products are designed and new markets opened up by successful integration of the discoveries of many diverse branches of basic science. This requires careful specification of interfaces, which should be tolerant to variation in environmental parameters, and cost-effective for a range of applications.

In a new scientific discipline, or one which has expanded too fast for its own good, it is a slow process to establish a consensus on what is the appropriate subdivision of the subject into its branches, and what are the appropriate methods of research within each branch. Exploration of the structure of the discipline and elucidation of the interfaces between its branches are necessary conditions of progress; and, of course, mathematical concepts, calculations and proofs play the same central role as they have in all well-established scientific disciplines.

In engineering methodology, two directions of interfacing can be distinguished:

1. Horizontal integration between components of a complex product, perhaps implemented in differing materials or technologies;

2. Vertical integration between levels of abstraction in the design process, ranging from requirements through specifications, designs, and ultimate implementation.

The scientific study of both kinds of interface can help not only to clarify the subject matter and structure of a scientific discipline; it can also help the engineer to improve product reliability and reduce time to market by avoiding the most insidious and most expensive kinds of error, those that lurk in the interfaces between components and between phases of the design. The benefits are even greater if the engineering calculations can be carried out or at least checked with the assistance of a computer.

That is the philosophical background to the safemos project, whose results are reported in this book, and of several related projects in other leading centres of research. They concentrate on what are recognized as issues central to computing science, including requirements, specifications, designs, programs, compilers, machines architectures, and logic design of hardware. Many of these interfaces are well understood; and here the project has aimed at an increase in rigour of formalization, preparing the ground for reliable mechanical support.

The safemos project concentrates on the most urgent problems of ensuring the reliability of designs and programs for embedded systems working in real-time; it is not aimed at any particular product, but it has clarified the principles of reliable design and implementation.

These principles, we hope, will be just as effective in the timely and reliable implementation of more general systems, where safety is not such a critical issue. But above all, the principles enlarge our basic scientific understanding of computing science, in a way that illuminates the structure of the whole subject and its methods of research.

C.A.R. Hoare

Preface

As the complexity of embedded computer-controlled systems increases, the present industrial practice for their development gives cause for concern, especially for safety-critical applications where human lives are at stake. The use of software in such systems has increased enormously in the last decade. Formal methods, based on firm mathematical foundations, provide one means to help with reducing the risk of introducing errors during specification and development. There is currently much interest in both academic and industrial circles concerning the issues involved, but the techniques still need further investigation and promulgation to make their widespread use a reality.

This book presents some results of research into techniques to aid the formal verification of mixed hardware/software systems. Aspects of system specification and verification from requirements down to the underlying hardware are addressed, with particular regard to real-time issues. The work presented is largely based around the Occam programming language and Transputer microprocessor paradigm. The HOL theorem prover, based on higher order logic, has mainly been used in the application of machine-checked proofs.

The book describes research work undertaken on the collaborative UK DTI/SERC-funded Information Engineering Directorate safemos project. The partners were Inmos Ltd, Cambridge SRI, the Oxford University Computing Laboratory and the University of Cambridge Computer Laboratory, who investigated the problems of formally verifying embedded systems. The most important results of the project are presented in the form of a series of interrelated chapters by project members and associated personnel. In addition, overviews of two other ventures with similar objectives are included as appendices.

The material in this book is intended for computing science researchers and advanced industrial practitioners interested in the application of formal methods to real-time safety-critical systems at all levels of abstraction from requirements to hardware. In addition, Chapters 1 and 11 contain material of a more general nature which may be of interest to managers in charge of projects applying formal methods, especially for safety-critical systems, and others who are considering their use.

In Part I of the book, Chapter 1 provides an introduction to the setting to which the rest of the book is intended to contribute, with particular regard to safety-critical systems, where correctness is of paramount importance. Standards are likely to provide a major motivating force for the use of formal methods in the development of such systems, and a selection of these are surveyed. Chapter 2 continues by giving an overview of the work undertaken on the safemos project, with a little more detail devoted to areas not covered in subsequent chapters.

Part II provides an introduction to the main theorem proving tool used on the safemos project (HOL) in Chapter 3, together with an example of how it may be used in modelling real-time systems in Chapter 4. Chapter 3 is included to give the reader not acquainted with the HOL mechanical theorem proving system a knowledge of its capabilities that will aid the reading of the rest of the book from Chapter 4 to 9.

Chapter 4 considers the mechanization of *timed transitions systems* (TTS) in HOL to allow modelling and reasoning about real-time systems. A traffic light controller example is used to present the principles involved. A mechanical proof environment could be further developed along the principles presented here to allow the specification and verification of real-time systems at a rather higher level of abstraction than considered in Part III. Embedding of requirements and design specifications, and techniques for demonstrating that a design meets its requirements using TTS proof rules are discussed.

Part III presents the use of HOL for developing and compiling software. Chapter 5 presents a complete self-contained case study of the verification of a small example program. The technique described is intended to be applied when the highest level of integrity is required. The timing aspects are modelled at the level of the machine clock cycle for the compiled object code. This is the only way to ensure completely accurate reasoning about the timing properties of the program. Of course this limits the size of code that can be handled tractably, but it is envisaged that small sections of safety-critical code could be verified in this manner to give the highest degree of confidence. The process is mechanized in HOL to help avoid human error and make it usable for non-trivial examples.

In the past, safety-critical software has often been developed using assembler programs due to the unreliability of high-level languages, and their unpredictable timing properties. On the **safemos** project, a small real-time Occam-like language and its compilation to a Transputer-like instruction set have been developed and mechanized in HOL. The language and its *interval temporal logic* semantics are presented in Chapter 6. Its compilation and the verification of this process are presented in Chapter 7. It is intended that the development of more reliable compilation for real-time programming along these lines will enable higher-level programming techniques to be used for safety-critical systems with more confidence in the future.

Correct software must be run on correct hardware for overall system correctness. Therefore the formal development of both aspects of a software/hardware system is important. Part IV presents aspects of verifying hardware designs. Chapter 8 discusses techniques to design microprocessors in a generic manner. Chapter 9 presents the development of a simple (but realistic) Transputer-like processor. The verified hardware described could be used to run programs compiled by the technique previously presented in Chapter 7.

An interesting recent development is the possibility of compiling hardware in a similar manner to that which software is routinely compiled today. Chapter 10 gives a more speculative presentation of how hardware for safety-critical systems could be developed in the future. These techniques are still an active area of research at an early stage of development and there is potential for considerable progress. For example, there is growing interest in the area of hardware/software co-design, which typically involves the intervention of a design engineer to determine suitable tradeoffs between the use of software and hardware.

Finally in Part V, aspects of technology transfer from formal methods academic research to industrial application are addressed. For formal methods to be accepted, their use must be integrated into current best industrial practice. It is too risky and expensive to completely replace existing methods. Chapter 11 discusses some of the issues involved and considers the future prospects for methods such as those investigated by the **safemos** project.

Two appendices present related work with similar aims to **safemos**, although using different techniques. At Computational Logic, Inc. (CLI) in the US, the verification of a number of related software and hardware levels has been undertaken using the Boyer-Moore theorem prover. Appendix A presents this inspiring example, and also some of their more recent work. In Europe, the collaborative ESPRIT **ProCoS** project has investigated formal techniques from requirements down to machine code and how these relate to each other. Appendix B gives an overview of the achievements of the first phase of this research project. These efforts are still ongoing and further progress and results are expected.

A large bibliography is included at the end of the book for those interested in particular areas of the **safemos** project, and related work by other researchers in the field of software/hardware system verification. A number of relevant standards and other publicly available documents are also included.

J.P. Bowen

Contact Addresses

Editor

Jonathan Bowen
Oxford University Computing Laboratory
Programming Research Group
Wolfson Building
Parks Road
OXFORD OX1 3QD
England
Email: Jonathan.Bowen@comlab.ox.ac.uk
URL: http://www.comlab.ox.ac.uk/oucl/people/jonathan.bowen.html

Contributors

Juanito Camilleri
Department of Computer Studies
University of Malta
University Heights
Msida
Malta G.C.
Email: juan%panther@carla.dist.unige.it

Rachel Cardell-Oliver
Department of Computer Science
University of Essex
Colchester
ESSEX CO4 3SQ
England
Email: cardr@essex.ac.uk

Mike Gordon
University of Cambridge
Computer Laboratory
New Museums Site
Pembroke Street
CAMBRIDGE CB2 3QG
England
Email: `mjcg@cl.cam.ac.uk`

Roger Hale
SRI International
Cambridge Research Centre (CRC)
Suite 23
Millers Yard
Mill Lane
CAMBRIDGE CB2 1RQ
England
Email: `rwsh@cam.sri.com`

John Herbert
SRI International
Cambridge Research Centre (CRC)
Suite 23
Millers Yard
Mill Lane
CAMBRIDGE CB2 1RQ
England
Email: `jmjh@cam.sri.com`

Prof. C.A.R. Hoare
Oxford University Computing Laboratory
Programming Research Group
Wolfson Building
Parks Road
OXFORD OX1 3QD
England
Email: `Tony.Hoare@comlab.ox.ac.uk`

He Jifeng
Oxford University Computing Laboratory
Programming Research Group
Wolfson Building
Parks Road
OXFORD OX1 3QD
England
Email: `Jifeng.He@comlab.ox.ac.uk`

Prof. Hans Langmaack
Christian-Albrechts Universität zu Kiel
Institut für Informatik und Praktische Mathematik
Preußerstraße 1–9
D-24105 Kiel
Germany
Email: `hl@informatik.uni-kiel.d400.de`

Ian Page
Oxford University Computing Laboratory
Programming Research Group
Wolfson Building
Parks Road
OXFORD OX1 3QD
England
Email: `Ian.Page@comlab.ox.ac.uk`

Paritosh Pandya
Tata Institute of Fundamental Research
Computer Science Group
TIFR, Homi Bhabha Road
Colaba
BOMBAY 400 005
India
Email: `pandya@tcs.tifr.res.in`

Andrew M. Pitts
University of Cambridge
Computer Laboratory
New Museums Site
Pembroke Street
CAMBRIDGE CB2 3QG
England
Email: `Andrew.Pitts@cl.cam.ac.uk`

Anders P. Ravn
Department of Computer Science
Technical University of Denmark
Building 344Ø
DK-2800 Lyngby
Denmark
Email: `apr@id.dth.dk`

David Shepherd
INMOS Limited
1000 Park Avenue
Aztec Way
Almondsbury
Bristol
AVON BS12 4SQ
England
Email: `des@inmos.co.uk`

Victoria Stavridou
Department of Computer Science
Royal Holloway and Bedford New College
University of London
Egham Hill
Egham
SURREY TW20 0EX
England
Email: `victoria@dcs.rhbnc.ac.uk`

Bill Young
Computational Logic, Inc.
1717 West Sixth Street
Suite 290
Austin
Texas 78703-4776
U.S.A.
Email: `young@CLI.com`