A SAFETY LICENSABLE COMPUTING ARCHITECTURE

This page is intentionally left blank

A SAFETY LICENSABLE COMPUTING ARCHITECTURE

Wolfgang A. Halang

FernUniversität

Soon-Key Jung

Chung Buk National University

Bernd J. Krämer

FernUniversität

Johan J. Scheepstra

University of Groningen

World Scientific

Singapore • New Jersey • London • Hong Kong

Published by

World Scientific Publishing Co. Pte. Ltd.

P O Box 128, Farrer Road, Singapore 9128

USA office: Suite 1B, 1060 Main Street, River Edge, NJ 07661

UK office: 73 Lynton Mead, Totteridge, London N20 8DH

A SAFETY LICENSABLE COMPUTING ARCHITECTURE

Copyright © 1993 by World Scientific Publishing Co. Pte. Ltd.

All rights reserved. This book, or parts thereof, may not be reproduced in any form or by any means, electronic or mechanical, including photocopying, recording or any information storage and retrieval system now known or to be invented, without written permission from the Publisher.

For photocopying of material in this volume, please pay a copying fee through the Copyright Clearance Center, Inc., 27 Congress Street, Salem, MA 01970, USA.

ISBN 981-02-1628-9

Contents

List of Figures	xi
List of Tables	xv
Authors	xvii
Foreword	xxi
Abstract	xxiii
Acknowledgement	xxv
1 Introduction	1
1.1 The Industrial Need	1
1.1.1 Reliability, Safety and Economic Efficiency	2
1.1.2 State of the Art in Developing Safety Related Systems	4
1.2 Boundary Conditions for the Problem Solution	5
1.3 Outline of the Solution	7
2 State of the Art in High Integrity Computing	9
2.1 Hardware	10
2.1.1 Types of Failures	10
2.1.2 Measures to Prevent Faults in Hardware Development	13

2.1.3	Measures to Achieve Fault Tolerance	14
2.1.4	Structural Types of Safety Related Systems	14
2.1.4.1	Hardware Redundancy	15
2.1.4.2	One-Channel Structures	17
2.1.4.3	Two-Channel Structures	17
2.1.4.4	Multi-Channel Structures	18
2.1.5	Detailed Safety Measures for Hardware Components	18
2.1.5.1	CPU	19
2.1.5.2	RAM/ROM	20
2.1.5.3	Buses	22
2.1.5.4	Input/Output Devices and Interface Hardware	23
2.1.5.5	Supervision of Program Execution	25
2.2	Software	28
2.2.1	System Design for Safety Related Software	28
2.2.1.1	Fault Prevention	28
2.2.1.2	Fault Tolerance	31
2.2.2	Measures for the Development of Safety Related Software	35
2.2.2.1	Software Quality Assurance	35
2.2.2.2	Structure of Development Projects	37
2.2.2.3	Software Requirements Specification . . .	38
2.2.2.4	Software Design and Coding	39
2.2.2.5	Software Diversity	43
2.2.3	Software Verification	46
2.2.3.1	Verification Process	46
2.2.3.2	Verification Plan	46

2.2.3.3	Verification Techniques	47
2.2.3.4	Software Requirements Verification . . .	51
2.2.3.5	Software Design Verification	51
2.2.3.6	Module and Code Verification	52
2.2.3.7	Hardware/Software Integration Verifica- tion	53
2.2.3.8	Safety Licensing	55
3	Programmable Logic Controllers	59
3.1	Description and Mode of Operation	59
3.2	Software Formulation and Development	60
3.3	Sequential Function Charts	69
4	A Safety Licensable Computing System	79
4.1	Application Specific Firmware	79
4.1.1	Automation of Chemical Processes	79
4.1.2	Emergency Shut-Down Systems	83
4.1.3	A Software Engineering Paradigm	90
4.2	Application Software Development and Verification	91
4.3	Hardware	93
4.4	Temporal Predictability	95
5	Detailed Hardware Description	99
5.1	Safety Oriented Architecture	99
5.2	Master Processor	100
5.3	FIFO-Queues and Comparators	104
5.4	System Clock	104
5.5	Input and Output	106

5.6	Architectural Overview	106
6	Rigorous Firmware Development	109
6.1	Specification Based Function Block Development	110
6.2	Formalising Requirements	114
6.2.1	Problem Statement	115
6.2.2	Requirements Specifications	117
6.2.3	OBJ	117
6.2.4	Functional Requirements of <code>nonlin</code>	119
6.3	Design Specification of Function Block <code>nonlin</code>	122
6.4	Specification Verification	125
6.4.1	Inductive Proofs with Equations	126
6.4.2	Correctness of Function <code>f'</code>	127
6.5	Specification Testing	130
6.6	Program Verification	133
6.6.1	A Subset of Structured Text: Simple ST	134
6.6.2	Weakest Preconditions	137
6.6.3	Correctness of Program <code>nonlin</code>	139
6.7	A Second Example	145
6.7.1	Problem Statement	145
6.7.2	Functional Requirements of <code>check</code>	146
6.7.3	Design of <code>check</code>	149
6.7.4	Correctness of Functions <code>cond</code> , <code>ua</code> , and <code>1a</code>	149
6.7.5	ST Program Implementing Function Block <code>check</code>	155
6.7.6	Hoare Style Proof Rules	157
6.7.7	Correctness of Program <code>check</code>	158
6.7.8	Improvement of Program <code>check</code>	166

6.8	Verification of a Timer Program using HOL	167
6.8.1	Abstract Syntax of Simple ST	169
6.8.2	Semantics of Simple ST in HOL	170
6.8.3	Timer Verification	172
6.9	Summary	177
7	High Integrity Application Software	181
7.1	Survey on a Software Development Environment	181
7.2	Detailed Description of the Tool Set	182
7.3	Verification Process	188
7.4	A Worked Example	189
8	Conclusion	199
8.1	Achieved Progress	199
8.2	Outlook	200
	Bibliography	203
A	Formal Hardware Definition in KARL III	215
A.1	Master Layout	215
A.2	Some Implementation Details	217
A.3	KARL III Source Code	224
	Index	245

This page is intentionally left blank

List of Figures

2.1	Safety proof of software according to DIN VDE 0116 . . .	56
2.2	System verification and validation	57
3.1	A sequential function chart	63
3.2	Function blocks with 3 or 2 levels of detail according to IEC SC65A	64
3.3	A graphically formulated program	70
3.4	Examples for elementary control sequences	71
3.5	Parallel control sequences	73
3.6	Some types of actions	76
3.7	Some further types of actions	77
4.1	The symbols representing And, Or, and Not	85
4.2	The graphical symbol for the timer	86
4.3	The symbols used for an input and an output	87
4.4	An example of a FLD	88
4.5	An example of a FLD (with dyadic Boolean operators only)	89
5.1	Configuration of a PLC with master/slave processors . . .	101
5.2	Instruction format	102

5.3	Comparison of FIFO data	105
5.4	Basic timing diagram	106
5.5	Output port and input port	107
5.6	Fault detecting master/slave PLC architecture	108
6.1	Rigorous function block development process	113
6.2	Non-linear function and its polygon approximation	116
6.3	Illustration of the recursive approximation algorithm defined in object <code>fpolygon</code>	123
6.4	Approximation algorithm <i>A</i>	140
6.5	Algorithm <i>A</i> decorated with weakest preconditions	141
6.6	Abstract functionality of function block check	150
6.7	Structure of program <i>S</i> computing COND and OLDC	160
6.8	Proof tree for program <i>S</i>	162
6.9	Simpler program derived from postcondition <i>Q</i>	167
7.1	A CASE tool set	183
7.2	Symbol classes used in graphical programming	186
7.3	FBD representation of a pressure control and supervision program	191
7.4	Net list representation of the example program	192
7.5	Net list representation of the example program (continued)	193
7.6	Net list representation of the example program (continued)	194
7.7	Structured Text representation of the example program	195
7.8	Machine code representation of the example program	196
7.9	Machine code representation of the example program (continued)	197
A.1	Timing diagram of the code control cell	217

A.2	Timing diagram of the address control cell	219
A.3	Timing diagram of step_block1	220
A.4	Layout of the master cell	221
A.5	Complete system layout	223

This page is intentionally left blank

List of Tables

2.1	Memory failure types and test tools	22
2.2	Program execution failure types and causes	25
6.1	Parameters of function block <code>nonlin</code>	115
6.2	States of function block <code>check</code>	150
A.1	Register decoding	216

This page is intentionally left blank

Authors

Wolfgang A. Halang, born 1951 in Essen, Germany, studied mathematics and theoretical physics at Ruhr-Universität Bochum, Germany. He received a doctorate with a thesis on numerical analysis in 1976. Subsequently, he studied computer science at Universität Dortmund concluding with a dissertation on function oriented structures for process control computers.

Simultaneously with the latter, he worked with Coca-Cola GmbH in Essen where he was in charge of real time applications in the chemical Research and Development Department. From 1985 until 1987, he was a professor for Systems Engineering at the King Fahd University of Petroleum & Minerals in Dhahran, Saudi Arabia, teaching courses mainly on real time and computer control systems. During the academic year 1987/88 he was a visiting (research) professor with the Department of Electrical and Computer Engineering and with the Computer Systems Group of the Coordinated Science Laboratory at the University of Illinois at Urbana-Champaign. From June 1988 till May 1989 he was back in Germany working in the process control division of Bayer AG. In June 1989 he became a full professor at the University of Groningen in the Netherlands holding the chair for applications-oriented computing science in the Department of Computing Science, which he headed between May 1990 and October 1991. Since 1992 he holds the chair for information technology in the Department of Electrical Engineering at FernUniversität Hagen in Germany.

His research interests comprise all major areas of hard real time systems with special emphasis on innovative and function oriented architectures and on application specific peripheral components for process control computers. Presently, he concentrates his interest on predictable system behaviour and on rigorous software verification and safety licensing.

He is the founder and European editor-in-chief of "Real-Time Systems", the international journal of time critical computing systems, director of the 1992 NATO Advanced Study Institute on Real Time Computing, has authored 3 books and over 100 refereed book chapters, journal publications and conference contributions, has given many guest

lectures on topics in this area, and is active in various professional organisations and technical committees as well as involved in the programme committees of many conferences.

Soon-Key Jung, born in Chinchun, Korea, in 1944, received a B.A. degree from Korea University, Seoul, in 1971 and a master's degree in computer science from Universität Dortmund, Germany, in 1982.

Since 1985 he is with the Department of Computer Engineering of Chung Buk National University in Cheongju, Korea, where he is currently an associate professor. He is also the director of his university's computer centre since 1992.

Before joining Chung Buk National University, Jung was manager of the System Engineering Department, DACOM Research and Development Institute in Seoul, Korea. In 1990 and 1991 he spent 16 months with the Department of Computing Science, University of Groningen, The Netherlands, carrying out research towards his doctorate. During this time and the following year he prepared a thesis entitled "An Architecture Supporting Safety Critical Software", which was revised and extended to this book.

His research interests are software engineering, real time computing, and database systems.

Bernd J. Krämer was born in Berlin, Germany, in 1947. Having started his professional career as an electrical engineer, he graduated in 1975 in computer science at Technical University of Berlin, from where he also obtained his doctorate in computer science.

He is now a professor and holds the chair of data processing technology at FernUniversität Hagen. He is also director of the Institute for New Technologies in Electrical Engineering at FernUniversität.

From 1975 to 1992 he was a senior researcher and international project leader at the German National Research Institute of Computer Science (GMD). From May 1989 to August 1990 he was an adjunct research professor at Naval Postgraduate School (NPS) in Monterey, California, supported by major grants from NSF and ONR. Prior to GMD he was a research associate at the Computing Centre of the City of Berlin.

His research interests are currently in formal specification, design, and analysis techniques for distributed software, hardware-software co-design, as well as system development methods and tools for high integrity software, especially as used in engineering applications. He was (co-) principal investigator of several international projects including the ESPRIT project 125 GRASPIN, a German government funded project on software process technology jointly performed with McGill University in Montreal, Canada, and an NSF sponsored project with NPS on a distributed support environment for prototyping real time applications. Currently he chairs a collaborative programme between Australian and German research groups in software engineering, and is a member of both the ESPRIT Basic Research Action IS-CORE and an international research group on software productivity at the Research Institute of Software Engineering, which is supported by the Japanese Ministry of Trading and Industry.

He is author and co-author of two books and numerous book chapters and articles in international journals and conference proceedings including IEEE TSE and Software, IEE SEJ, JSS, ICSE, IWSSD, COMPSAC, and SAFECOMP. He has designed two novel specification languages for distributed applications and programming environments design. He served on many programme committees including ICICIS, CASE, and I-SEE. He is a member of the ACM, the Gesellschaft für Informatik, the technical committee 3.3 (communication and distributed systems) of GI, and the German standardisation committee DIN NI-21 (Basic Reference Model on Open Distributed Processing).

Johan J. Scheepstra, born 1962 in Groningen, The Netherlands, studied computing science with specialisation in real time systems at the University of Groningen, where he received his master's degree in 1992.

The topic of his master's thesis was the development of a new concept for an emergency shut-down system based on a fail safe programmable logic controller as described in Chapter 4 of this book. He showed the feasibility of this approach by building an operational prototype.

During the winter 1991/92 he spent 4 months at the Faculty of Engineering, Hiroshima University, Japan, carrying out theoretical research on industrial safeguarding. This stay was supported by a scholarship of the Dutch Government.

This page is intentionally left blank

Foreword

The user of any computer system needs assurance of the performance and reliability of the system. In the present day applications of Information Technology there are two areas of particular concern. Firstly, the very large banking, stock market and financial services networks, and secondly the "embedded systems" used to control vital transportation systems or continuous flow production plants. By "embedded systems" we mean those applications of computer hardware and software in which the equipment is hidden from public view, within the electronics bays of aircraft, deep inside the protection mechanisms of nuclear reactors, within the engine compartments of cars and trucks, and in hundreds of similar real time control systems.

Many of these regimes involve the independent certification of the plant or vehicle before operation or use is attempted. No nation will allow a civil aircraft to fly without a Certificate of Airworthiness and no nuclear reactor is allowed to operate without independent assessment of its design, protection and operating procedures.

The work reported in this book fits into the category of such embedded systems. Based on the concepts introduced by the VIPER research programme in the United Kingdom and the parallel research on the FM8502 microprocessor in the U.S.A., the authors have produced an architecture which is capable potentially of achieving some of the industrial and social needs for high integrity computing outlined in this Foreword. Much development remains to be completed, but in this careful research and documentation there are the seeds from which future industrial techniques may grow.

Coventry, June 1993

W J Cullyer

This page is intentionally left blank

Abstract

The licensing authorities are very reluctant in approving safety related systems, whose behaviour is exclusively program controlled. In general, safety licensing is still denied for highly safety critical systems relying on software with non-trivial complexity. There are already a number of established methods and guidelines, which have proven their usefulness for the development of high integrity software employed for the control of safety critical technical processes. Prior to its application, such software is further subjected to appropriate measures for its verification and validation. However, according to the present state of the art, these measures cannot guarantee the correctness of larger programs with mathematical rigour.

In order to provide a remedy for this unsatisfactory situation, it is the purpose of this book to develop a special — and necessarily simple — computer system, which can carry out safety related functions within the framework of distributed process control systems or programmable logic controllers (PLCs). According to well established and accepted methods, the computer hardware is designed to work correctly and to be fault detecting. It could be based on the VIPER [75] chip, the correctness of whose design has been formally proven. More emphasis is given to the software side, since it is felt that software dependability still needs to catch up with the one already achieved by hardware. The presented approach is unique as being the first effort to provide support for software verification already in the architecture.

The architecture features full temporal predictability, determinism, and supervision of program execution and of all other activities of the computer system. There is explicit support for the software verification method of diverse back translation and for sequential control, since many automation programs including safety relevant tasks are of that kind. A study in chemical industry has revealed, that all process control programs can be constructed from a small set (some 70) of function modules (“software ICs”), whose correctness can be formally proven due to their limited complexity. This observation holds for other industrial application areas, too. We present a new software engineering paradigm based on such libraries of application specific sets of modules, whose

object code, after rigorous verification, must be frozen in ROMs. It consists of graphically formulating safety related automation programs by interconnecting module instances with an appropriate CAD tool. The verification of compilers transforming graphical software representations into object code is still impossible — but also not necessary, because only the module interconnections need to be verified for application programs. For this task the architecturally supported method of back translation can be employed very efficiently and cost effectively.

Acknowledgement

The authors are deeply indebted to Messrs. M.A. Berkenbosch and H.G.J.M. van Aartsen for the detailed formal description of the hardware designed in this book. Their work is presented in the Appendix.

Furthermore, the authors would like to express their appreciation to Mr. N. Völker for working out the correctness proof for a timer function block on the basis of predicate calculus. This proof is given in Chapter 6. Mr. I. Scheler was helpful in preparing some of the illustrations in the same chapter and in designing the title page.