

Contents

Preface	xi
Chapter 1	
Hardware-dependent Software—Introduction and Overview	1
<i>Wolfgang Ecker, Wolfgang Müller and Rainer Dömer</i>	
1.1. Increasing Complexity	2
1.2. Hardware-dependent Software	6
1.3. Chapter Overview	10
References	13
Chapter 2	
Basic Concepts of Real Time Operating Systems	15
<i>Franz Rammig, Michael Ditzte, Peter Janacik, Tales Heimfarth, Timo Kerstan, Simon Oberthuer and Katharina Stahl</i>	
2.1. Introduction	16
2.2. Characteristics of Real-Time Tasks	17
2.3. Real-Time Scheduling	20
2.4. Operating System Designs	25
2.5. RTOS for Safety Critical Systems	31
2.6. Multi-Core Architectures	34
2.7. Operating Systems for Wireless Sensor Networks	37
2.8. Real-Time Requirements of Multimedia Application	40
2.9. Conclusions	42
References	44
Chapter 3	
UEFI: From Reset Vector to Operating System	47
<i>Vincent Zimmer, Michael Rothman and Robert Hale</i>	
3.1. Introduction	48
3.2. The Ever Growing Ever Changing BIOS	48
3.3. Time for a Change	51

3.4. UEFI and Standardization of BIOS	52
3.5. Framework, Foundation, and Platform Initialization	59
References	66
Chapter 4	
Hardware Abstraction Layer—Introduction and Overview	67
<i>Katalin Popovici and Ahmed Jerraya</i>	
4.1. Introduction	68
4.2. Software Stack	70
4.3. Hardware Abstraction Layer	74
4.4. Existing Commercial HAL	78
4.5. Overview of the Software Design and Validation Flow	80
4.6. HAL Execution and Simulation Using Software Development Platforms	83
4.7. Experiments	87
4.8. Conclusions	91
References	92
Chapter 5	
HW/SW Interface—Implementation and Modeling	95
<i>Wolfgang Ecker, Volkan Esen, Thomas Steininger and Michael Velten</i>	
5.1. Introduction	96
5.2. Reading and Writing Data Words	97
5.3. Bit Fields	104
5.4. Register Address and Data Mismatch	113
5.5. Textual Specification of the SIF	121
5.6. Register Header File	127
5.7. SIF Driver Functions	131
5.8. Synchronization	135
5.9. Template Based Code Generation	137
5.10. Modeling the HW/SW Interface	141
5.11. Conclusions	148
References	149
Chapter 6	
Firmware Development for Evolving Digital Communication Technologies	151
<i>Stefan Heinen and Michael Joost</i>	
6.1. Introduction	152
6.2. Evolution of Wireless Standards and the Consequences	153
6.3. System Level Design Flow	155
6.4. Hardware / Firmware Interface	161
6.5. Test Bench	165
6.6. Summary	171
References	171

Chapter 7

Generation and Use of an ASIP Software Tool Chain	173
<i>Sterling Augustine, Marc Gauthier, Steve Leibson, Peter Macliesh, Grant Martin, Dror Maydan, Nenad Nedeljkovic and Bob Wilson</i>	
7.1. Introduction	174
7.2. Range of Processor Configurability	175
7.3. Models for Generating Software Development Tools	176
7.4. Evolution of Tool-Development Approaches	179
7.5. The C/C++ Compiler	183
7.6. The Assembler	186
7.7. The Linker	188
7.8. The Loader	190
7.9. The Disassembler	191
7.10. The Debugger	192
7.11. Other Software-Development Tools	192
7.12. Operating Systems and Other System Software	192
7.13. The Instruction Set Simulator (ISS)	194
7.14. System Simulation	196
7.15. The IDE (Integrated Development Environment)	197
7.16. Conclusions and Futures	201
References	202

Chapter 8

High-Level Development, Modeling and Automatic Generation of Hardware-Dependent Software	203
<i>Gunar Schirner, Rainer Dömer and Andreas Gerstlauer</i>	
8.1. Introduction	204
8.2. Software-enabled System Design Flow	208
8.3. Software Generation Overview	210
8.4. Hardware-dependent Software Generation	211
8.5. Experimental Results	223
8.6. Conclusions	228
References	229

Chapter 9

Accurate RTOS Modeling and Analysis with SystemC	233
<i>Henning Zabel, Wolfgang Müller and Andreas Gerstlauer</i>	
9.1. Introduction	234
9.2. SystemC RTOS Model	240
9.3. Related Approaches	252
9.4. Applications	254
9.5. Conclusions	258
References	259

Chapter 10

Verification of AUTOSAR Software by SystemC-Based Virtual Prototyping	261
<i>Matthias Krause, Oliver Bringmann and Wolfgang Rosenstiel</i>	
10.1. Introduction	262
10.2. Concepts of AUTOSAR	264
10.3. Different System Views on Distributed Embedded Systems	269
10.4. Applying SystemC for AUTOSAR Software Verification	273
10.5. Integration of Timing Behavior into Virtual Prototypes	283
10.6. Application Example	286
10.7. Conclusions	290
References	291
Index	295