HARDWARE-DEPENDENT SOFTWARE
PRINCIPLES AND PRACTICE

Edited by
WOLFGANG ECKER
Infineon Technologies AG, Germany

WOLFGANG MÜLLER
Paderborn University, Paderborn, Germany

RAINER DÖMER
University of California, Irvine, U.S.A.

Kluwer Academic Publishers
Boston/Dordrecht/London

As the editors spend considerable time for the preparation of this book, they would like to thank their families for their support.

Wolfgang Ecker dedicates this book to his wife Monika and children Johannes, Stephanie, and Matthias.

Wolfgang Müller dedicates this book to his wife Barbara and children Philipp, Maximilian, and Tabea.

Rainer Dömer dedicates this book to his wife Julia and children Sophie, Klara, and Simon.

Contents

Preface ix

Chapter 1

Hardware-dependent Software – Introduction and Overview 1
Wolfgang Ecker, Wolfgang Müller and Rainer Dömer
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
Franz Rammig, Michael Ditze, Peter Janacik, Tales Heimfarth, Timo Kerstan, Simon Oberthuer and Katharina Stahl
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 45

Chapter 3

UEFI: From Reset Vector to Operating System 47
Vincent Zimmer, Michael Rothman and Robert Hale
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
Katalin Popovici and Ahmed Jerraya
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 90
References 93

Chapter 5

HW/SW Interface – Implementation and Modeling 97
Wolfgang Ecker, Volkan Esen, Thomas Steininger and Michael Velten
5.1. Introduction 98
5.2. Reading and Writing Data Words 99
5.3. Bit Fields 106
5.4. Register Address and Data Mismatch 115
5.5. Textual Specification of the SIF 123
5.6. Register Header File 129
5.7. SIF Driver Functions 133
5.8. Synchronization 137
5.9. Template Based Code Generation 139
5.10. Modeling the HW/SW Interface 143
5.11. Conclusions 150
References 151

Chapter 6

Firmware Development for Evolving Digital Communication Technologies 153
Stefan Heinen and Michael Joost
6.1. Introduction 154
6.2. Evolution of Wireless Standards and the Consequences 155
6.3. System Level Design Flow 157
6.4. Hardware / Firmware Interface 163
6.5. Test Bench 167
6.6. Summary 172
References 173

Chapter 7

Generation and Use of an ASIP Software Tool Chain 175
Sterling Augustine, Marc Gauthier, Steve Leibson, Peter Macliesh, Grant Martin, Dror Maydan, Nenad Nedeljkovic and Bob Wilson
7.1. Introduction 176
7.2. Range of Processor Configurability 177
7.3. Models for Generating Software Development Tools 178
7.4. Evolution of Tool-Development Approaches 181
7.5. The C/C++ Compiler 185
7.6. The Assembler 188
7.7. The Linker 190
7.8. The Loader 192
7.9. The Disassembler 193
7.10. The Debugger 193
7.11. Other Software-Development Tools 194
7.12. Operating Systems and Other System Software 194
7.13. The Instruction Set Simulator (ISS) 195
7.14. System Simulation 198
7.15. The IDE (Integrated Development Environment) 199
7.16. Conclusions and Futures 203
References 204

Chapter 8

High-Level Development, Modeling and Automatic Generation of Hardware-Dependent Software 205
Gunar Schirner, Rainer Dömer and Andreas Gerstlauer
8.1. Introduction 206
8.2. Software-enabled System Design Flow 210
8.3. Software Generation Overview 212
8.4. Hardware-dependent Software Generation 213
8.5. Experimental Results 225
8.6. Conclusions 231
References 232

Chapter 9

Accurate RTOS Modeling and Analysis with SystemC 235
Henning Zabel, Wolfgang Müller and Andreas Gerstlauer
9.1. Introduction 236
9.2. SystemC RTOS Model 242
9.3. Related Approaches 254
9.4. Applications 256
9.5. Conclusions 260
References 261

Chapter 10

Verification of AUTOSAR Software by SystemC-Based Virtual Prototyping 263
Matthias Krause, Oliver Bringmann and Wolfgang Rosenstiel
10.1. Introduction 264
10.2. Concepts of AUTOSAR 266
10.3. Different System Views on Distributed Embedded Systems 271
10.4. Applying SystemC for AUTOSAR Software Verification 275
10.5. Integration of Timing Behavior into Virtual Prototypes 285
10.6. Application Example 288
10.7. Conclusions 292
References 294

Index 297

Preface

Hardware-dependent Software (HdS) plays a key role in desktop computers and servers for many years. Recently, the relevance of HdS in the domains of embedded systems and Systems-on-Chip (SoCs) has significantly increased, mainly due to its flexibility, the possibility of late change, and the quick adaptability.

Modern SoCs, on a single die integrated embedded systems, often contain multiple programmable cores, including general purpose processors, digital signal processors (DSPs), and/or application specific instruction set processors (ASIPs) requiring a large amount of low level software. Mobile phones and automotive control systems meanwhile come with complex boot loaders and include multiple communication protocol stacks of considerable size. Here and in many other application areas, the number and complexity of standards that need to be supported have steadily grown. For mobile phones, for instance, the set of currently expected standards includes GSM, GPRS, EDGE, UMTS, Bluetooth, TCP/IP, and IrDA, to only name a few.

In this context, HdS has become a crucial factor in embedded system design since it allows to accommodate and adapt late changes in the hardware platform as well as in the application software. Thus, even last minute changes can be quickly performed. On the other hand, changes in the HdS are often hard to track and can have a complex impact on the system with a potential for total system failure. HdS also critically influences the system performance and power management. Consequently, HdS must be carefully designed and maintained.

In contrast to its importance in the area of electronic systems design, the role of HdS is most often underestimated. Considering todays literature, we can only find very few introductory and application-oriented text books. To overcome this gap, we have brought together experts from different HdS areas in this book. By providing a comprehensive overview of general HdS principles, tools, and applications, we feel that this book provides adequate insight into the current technology and upcoming developments in the domain of HdS. The reader will find a text book with self-contained introductions to the principles of Real-Time Operating Systems (RTOS), the emerging BIOS successor UEFI,

and the Hardware Abstraction Layer (HAL). Further chapters cover industrial applications, verification, and tool environments.

This book would not have been possible without the help and contributions of many people. First of all, we would like to thank Mark de Jongh and Cindy Zitter from Springer Verlag who supported us throughout the publication process. We also thank the contributing authors for their great cooperation through the entire process. For the review of individual chapters and valuable comments and suggestions, we acknowledge the help of Stephen A. Edwards (Columbia University), Petru Eles (Linköpings Universitet), Andreas Gerstlauer (University of Texas, Austin), Grant Martin (Tensilica Inc.), and Graziano Pravadelli (Universita di Verona). Finally, we thank Christof Poth who provided us with the sparkling picture for our book cover.

Wolfgang Ecker
Infineon Technologies AG, Munich, Germany

Wolfgang Müller
Paderborn University, Paderborn, Germany

Rainer Dömer
University of California, Irvine, USA

Index

Abstract canonical RTOS model, 243
Abstract RTOS library, 243
Abstract RTOS model, 281
Access methods, 159
ACPI, 62
AcpiNvs, 63
Addressing
  byte, 117
  half-word, 117
  word, 117
Address translation, 108
Algorithmic performance, 157
Annotated segments, 240
Application binary interface, 26
Application layer, 70
Application software, 8–10, 31, 77, 81, 135, 268
Architectural Protocol (AP), 64
ARINC 653, 31
ARM, 62
ARTOS, 243
ASIC, 2
ASIP, 3, 12, 68, 175
ATA, 49
Atomic blocks, 240
AUTOSAR, 12, 256, 260, 264
Base address, 101
Basic block level, 240
Basic Software (BSW), 266
Basic task, 283
Best Case Execution Time (BCET), 287
BIOS, 10, 48–49, 51
BIOS Boot Specification (BBS), 65
Bit field, 106
  access control, 112
  C, 107
  external access, 126
  internal access, 126
  offset, 126
  readable, 107
  readable/writable, 111
  shadow variable, 109
  specification, 123
  structured approach, 115
  width, 126
  writable, 108
Block I/O, 55
Block-lockable flash, 66
Board Support Package (BSP), 78
Boot Device Selection (BDS), 65
Boot
  firmware, 10
  manager, 65
  order, 56
  ROM, 48
Bus bridge, 115
Bus Functional Model (BFM), 159, 238
Bus interface model, 160
Cache-As-RAM (CAR), 61
Call Admission Control (CAC), 41
CAN, 260
Canonical RTOS model, 240
Capsules, 62
Chip
  capacity, 4
  complexity, 4
COCOMO, 5
Code generation, 81, 153, 212–213, 266, 270
Code instrumentation, 286
Commercial HAL, 78
Communicating processes, 273
Communication, 268
  protocol stack, 10
  refinement, 279
Communication software component, 73
Compatibility Support Module (CSM), 59
Complex device drivers, 269
Complexity, 2–3
Component-based design, 6
Computation refinement, 279
Configuration, 7, 271
Context switch, 26, 72, 74, 76, 78, 205, 242–243, 249
Cooperative multitasking, 242
Critical section, 20
Cycle Accurate (CA), 237, 260
Cycle Callable (CC), 274
Deadline, 17, 258
Debugging, 83, 163, 236

Delta-cycle, 242
Dependency Expression (DEPEX), 63–64
Dereferencing operator, 108
Development time, 5
Device driver, 10, 75
Digital signal processor (DSP), 2, 68, 79, 87, 90, 177, 222, 238
DO-178B, 31
DRAM, 50
Driver Execution Environment (DXE), 60
DXE
  architectural protocols, 64
  core, 62
  IPL, 63
  modules, 62
Dynamic timing simulation, 287
Earliest Deadline First (EDF), 23–24, 241
Economics, 5
ECU
  abstraction, 268
  configuration, 270
  software architecture, 269
EDA, 8
EDA tools, 12
EDGE, 155
EFI Developer Kit (EDK), 53
Electronic Control Unit (ECU), 2, 227, 256–257, 264
Electronic design automation, 8
Electronic System Level (ESL), 210
Embedded software, 4
Embedded software design, 5
Embedded system, 3
Endianness, 118
  adopt, 119
  big, 119
  little, 119
  middle, 119
Engine management system, 258
Event-based kernel, 37
Event-driven simulation, 160
Evolution of mobile standards, 155
Executes-in-place (XIP), 61
Execution model, 85
Execution time line, 242
Extended task, 283
Extensible Firmware Interface (EFI), 47, 51
External communication, 214
Fabrication technology, 7
FAT, 56
FAT12, 56
FAT16, 56
FAT32, 56
Files, 62
Firmware, 3, 8, 11, 238
  costs, 8
  development, 12, 153
  file system, 62
Flexibility, 7
FlexRay, 260, 288
FLIX, 186, 189, 191
Formal specification, 164
Fuel injection control, 258
Globally Unique Identifier (GUID), 52, 64
Golden reference, 160
GPRS, 155
GSM, 155, 225
HAL APIs, 79
Hall sensors, 258
Hand-Off-Block (HOB), 62–63
Hardware Abstraction Layer (HAL), 10, 12, 64, 67, 73–74, 194
Hardware
  architecture, 78
  design gap, 4
  drivers, 71
  platform, 9
Hardware-firmware split, 158
Hardware-in-the-Loop (HIL), 256
Hardware-software
  interface, 12, 86
  simulation, 84
  split, 156
HdS
  architecture, 9
  layer, 9, 70–71
  topics, 11
Heterogeneous architectures, 68
Hierarchical bus, 115
High Speed Downlink Packet Access (HSDPA), 155
Human Interface Infrastructure (HII), 52
HW/SW interface
  memory mapped, 99
  register mapped, 100
  special instructions, 100
IA-32, 62
IEC 61508, 31
Instruction set architecture, 30
Instruction set simulation, 236
Instruction set simulator, 83, 162
Instrumentation, 286
Integrated Development Environment (IDE), 3, 175, 199, 256
Intellectual Property (IP), 4
Internal communication, 214
Inter Process Communication (IPC), 29, 35, 238
Interrupt-based Multi-tasking, 221
Interrupt-based synchronization, 217
Interrupt handler, 25, 73, 217–219, 224
Interrupt handling, 236
Interruptible time specification, 247
Interrupt request (IRQ), 26, 75–77, 139, 236
Interrupt Service Routine (ISR), 236, 251

I/O controller driver, 59
I/O operations, 73
ISCSI, 52
ISR scheduler, 248
Itanium, 62, 66
IUT, 168
Joint Source Channel Coding, 42
Kernel design, 25
KLoC, 5
Latency, 229
Layered organization, 70
Link level simulation, 158
Logical Block Addresses (LBA), 56
Logic analyzers, 236
LTE, 155
Machine Check Architecture (MCA), 66
Marshalling, 215
Meta model, 140
Microcontroller abstraction, 268
Micro Control Unit (MCU), 3
Microelectronics, 2
Microkernel, 28
Middleware, 10
MIMO, 155
Model-in-the-Loop (MIL), 256
Monolithic kernels, 27
Moore's law, 2, 4
Motion JPEG decoder, 87
MPSoC design flow, 69
μ-Itron, 240, 254
Multimedia application, 40
Multiprocessing, 71
Multiprocessor architectures, 34
Multi-processor system-on-a-chip (MPSoC), 255
Multi-rate system, 21
Nested interrupts, 249
Network bandwidth, 3
Nielson's law, 3
Non-maskable interrupts (NMI), 249
Non-volatile random access memory (NVRAM), 55
Operating System, 9–10, 72, 268
Orthogonalization, 156
OS boot loader, 51
OS loader, 55
PCI, 49
PCIe, 49
PE/COFF, 64
PEI Modules (PEIM), 62
PEIM-to-PEIM Interface (PPI), 63
PERFidiX, 255
Performance evaluation, 258
Periodic task, 18–21, 42, 251–252
Peripheral Component Interconnect (PCI), 64
PI boot, 60
Platform-based design, 8
Platform Initialization (PI), 59
Platform Management Interrupt (PMI), 66
Polling-based synchronization, 216
PORTOS, 254
POSIX, 240, 255
Power Management, 38
Power On Self Test (POST), 10, 48
PPort, 267
Pre-EFI Initialization (PEI), 60, 62
Preemptive scheduling, 72
Preemptive thread multitasking kernels, 38
Priority-based scheduler, 249
Processor core, 71, 87, 183, 243
Processor utilization factor, 22
Process technology, 7
Productivity, 4, 6
  crash, 6
  gap, 4
Programmers View (PV), 273
Project planning, 8
Pseudo-parallel, 242
Quality of Service (QoS), 40–42
Rate monotonic priority assignment, 21
Rate monotonic scheduling, 21, 241
Real mode, 50
Real-Time Operating System (RTOS), 16, 31, 220, 235, 283
Real-Time scheduling, 21, 72
Reconfiguration, 8
Reduced Instruction Set Computer (RISC), 3, 183
Register access
  class-based, 103
  C struct, 104
  function-based, 101
  functions, 131
  macro-based, 103
  object-based, 101
  structured approach, 115
Register
  addressable unit, 125
  auto-shadow, 120
  bit field structure, 129
  block transfer, 121
  file model, 159
  indexing bit fields, 121
  mirror size, 125
  offset, 125
  specification, 123
  types, 166
  width, 125
Regression, 163
Regression testing, 167
Response time, 258
Response time analysis, 23, 236
Reuse, 7
Round-Robin scheduling, 241, 249
RPort, 267
RTL, 160
RTOS

Abstraction Layer (RAL), 220
API, 254
context, 243
models, 239
simulation, 12, 235
state model, 244
Runnable Entities (RE), 267
Run Time Environment (RTE), 266
Safety-critical systems, 31
Scalability, 5
SCAS, 254
Schedulability, 22
Scheduler Synchronization, 246
Scheduling, 21–24, 72, 241, 243
Scheduling decisions, 72
SCSI, 49
Sections, 62
Security phase (SEC), 60
Segments, 245
Semaphore, 20, 73, 217, 249
Sequentialization, 243
Services, 268
SIF, 6, 123
driver functions, 6, 133
interrupt interface, 124
RX interface, 124
RX state machine, 124
TX interface, 124
TX state machine, 124
Simulation, 85, 241, 276
Simulation kernel, 242
Simulation speed trade-offs, 237
Simulation time, 245
Single source, 157
SMRAM, 66
Software
complexity, 3–4
component, 267
component template, 266
database, 224
design, 3, 7
design cost, 5
design flow, 80
design gap, 5
design productivity, 3, 5
development, 5
development platforms, 83
dominance, 7
generation, 12, 81, 207, 212
portability, 77
reuse, 69
stack, 9, 70–71
stack composition, 81
synthesis, 207
validation, 82
Software-in-the-Loop (SIL), 256
SpecC, 208, 254
Specification model, 210
Speedups, 238
Staff months, 5
Static timing analysis, 236, 287
Stimuli generators, 169
Stream-Driven simulation, 157
Synchronization, 73, 216, 243
interrupt, 138
polling, 138
System Abstraction Layer (SAL), 66
SystemC, 160, 198, 208, 235, 263
RTOS library, 235
simulation, 242
threads, 242
wait statements, 242, 246
System design gap, 5
SystemDesk (dSPACE), 256
System level
design, 208
design language, 208, 239
modeling, 157
System Management BIOS (SMBIOS), 64
System Management Bus (SMBUS), 50
System Management Mode (SMM), 66
System Management RAM (SMRAM), 66
System-on-Chip (SoC), 3
System Table (SST), 66
Target binary, 224
TargetLink (dSPACE), 256
Task concurrency management, 35
Task Control Block (TCB), 243
Task level, 238
Template engine, 142
Test bench, 163, 167
TIE, 177
Time-annotated software segments, 240
Time annotation, 259
Timed segments, 242
Time line, 242
Time specification, 246
Time-Triggered Protocol (TTP), 36
Timing behavior, 285
Trace, 170
Trace box, 236
Tracing hardware, 236
Transaction accurate architecture, 86
Transaction level, 160
Transaction Level Modeling (TLM), 144, 198, 208, 212, 238, 255
Transformation, 284
UEFI
drivers, 56
protocols, 54
UMTS, 155
Unified Extensible Firmware Interface (UEFI), 47, 52
USB, 49

Validation flow, 80
Virtual Functional Bus (VFB), 266
Virtual machines, 29
Virtual memory, 25
Virtual prototype, 84, 157, 162
Virtual prototyping, 265
VLIW, 177, 186
Volatile, 101
Volumes, 62
Wireless Sensor Network (WSN), 37
Worst Case Execution Time (WCET), 236, 287
Worst Case Response Time (WCRT), 236
WSNOS architecture, 38
X64, 62
X86, 60
XIP, 61, 63
XML, 159, 163, 257