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.
<table>
<thead>
<tr>
<th>Chapter 4</th>
<th>67</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware Abstraction Layer – Introduction and Overview</td>
<td></td>
</tr>
<tr>
<td><em>Katalin Popovici and Ahmed Jerraya</em></td>
<td></td>
</tr>
<tr>
<td>4.1. Introduction</td>
<td>68</td>
</tr>
<tr>
<td>4.2. Software Stack</td>
<td>70</td>
</tr>
<tr>
<td>4.3. Hardware Abstraction Layer</td>
<td>74</td>
</tr>
<tr>
<td>4.4. Existing Commercial HAL</td>
<td>78</td>
</tr>
<tr>
<td>4.5. Overview of the Software Design and Validation Flow</td>
<td>80</td>
</tr>
<tr>
<td>4.6. HAL Execution and Simulation Using Software Development Platforms</td>
<td>83</td>
</tr>
<tr>
<td>4.7. Experiments</td>
<td>87</td>
</tr>
<tr>
<td>4.8. Conclusions</td>
<td>90</td>
</tr>
<tr>
<td>References</td>
<td>93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 5</th>
<th>97</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW/SW Interface – Implementation and Modeling</td>
<td></td>
</tr>
<tr>
<td><em>Wolfgang Ecker, Volkan Esen, Thomas Steininger and Michael Velten</em></td>
<td></td>
</tr>
<tr>
<td>5.1. Introduction</td>
<td>98</td>
</tr>
<tr>
<td>5.2. Reading and Writing Data Words</td>
<td>99</td>
</tr>
<tr>
<td>5.3. Bit Fields</td>
<td>106</td>
</tr>
<tr>
<td>5.4. Register Address and Data Mismatch</td>
<td>115</td>
</tr>
<tr>
<td>5.5. Textual Specification of the SIF</td>
<td>123</td>
</tr>
<tr>
<td>5.6. Register Header File</td>
<td>129</td>
</tr>
<tr>
<td>5.7. SIF Driver Functions</td>
<td>133</td>
</tr>
<tr>
<td>5.8. Synchronization</td>
<td>137</td>
</tr>
<tr>
<td>5.9. Template Based Code Generation</td>
<td>139</td>
</tr>
<tr>
<td>5.10. Modeling the HW/SW Interface</td>
<td>143</td>
</tr>
<tr>
<td>5.11. Conclusions</td>
<td>150</td>
</tr>
<tr>
<td>References</td>
<td>151</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 6</th>
<th>153</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware Development for Evolving Digital Communication Technologies</td>
<td></td>
</tr>
<tr>
<td><em>Stefan Heinen and Michael Joost</em></td>
<td></td>
</tr>
<tr>
<td>6.1. Introduction</td>
<td>154</td>
</tr>
<tr>
<td>6.2. Evolution of Wireless Standards and the Consequences</td>
<td>155</td>
</tr>
<tr>
<td>6.3. System Level Design Flow</td>
<td>157</td>
</tr>
<tr>
<td>6.4. Hardware/Firmware Interface</td>
<td>163</td>
</tr>
<tr>
<td>6.5. Test Bench</td>
<td>167</td>
</tr>
<tr>
<td>6.6. Summary</td>
<td>172</td>
</tr>
<tr>
<td>References</td>
<td>173</td>
</tr>
</tbody>
</table>
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
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

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

Hardware-Dependent Software
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
<table>
<thead>
<tr>
<th>INDEX</th>
<th>299</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O controller driver, 59</td>
<td></td>
</tr>
<tr>
<td>I/O operations, 73</td>
<td></td>
</tr>
<tr>
<td>iSCSI, 52</td>
<td></td>
</tr>
<tr>
<td>ISR scheduler, 248</td>
<td></td>
</tr>
<tr>
<td>Itanium, 62, 66</td>
<td></td>
</tr>
<tr>
<td>IUT, 168</td>
<td></td>
</tr>
<tr>
<td>Joint Source Channel Coding, 42</td>
<td></td>
</tr>
<tr>
<td>Kernel design, 25</td>
<td></td>
</tr>
<tr>
<td>KLoC, 5</td>
<td></td>
</tr>
<tr>
<td>Latency, 229</td>
<td></td>
</tr>
<tr>
<td>Layered organization, 70</td>
<td></td>
</tr>
<tr>
<td>Link level simulation, 158</td>
<td></td>
</tr>
<tr>
<td>Logical Block Addresses (LBA), 56</td>
<td></td>
</tr>
<tr>
<td>Logic analyzers, 236</td>
<td></td>
</tr>
<tr>
<td>LTE, 155</td>
<td></td>
</tr>
<tr>
<td>Machine Check Architecture (MCA), 66</td>
<td></td>
</tr>
<tr>
<td>Marshalling, 215</td>
<td></td>
</tr>
<tr>
<td>Meta model, 140</td>
<td></td>
</tr>
<tr>
<td>Microcontroller abstraction, 268</td>
<td></td>
</tr>
<tr>
<td>Micro Control Unit (MCU), 3</td>
<td></td>
</tr>
<tr>
<td>Microelectronics, 2</td>
<td></td>
</tr>
<tr>
<td>Microkernel, 28</td>
<td></td>
</tr>
<tr>
<td>Middleware, 10</td>
<td></td>
</tr>
<tr>
<td>MIMO, 155</td>
<td></td>
</tr>
<tr>
<td>Model-in-the-Loop (MIL), 256</td>
<td></td>
</tr>
<tr>
<td>Monolithic kernels, 27</td>
<td></td>
</tr>
<tr>
<td>Moore’s law, 2, 4</td>
<td></td>
</tr>
<tr>
<td>Motion JPEG decoder, 87</td>
<td></td>
</tr>
<tr>
<td>Multi-processor system-on-a-chip (MPSoC), 255</td>
<td></td>
</tr>
<tr>
<td>Multi-programmer system, 21</td>
<td></td>
</tr>
<tr>
<td>Nested interrupts, 249</td>
<td></td>
</tr>
<tr>
<td>Network bandwidth, 3</td>
<td></td>
</tr>
<tr>
<td>Nielson’s law, 3</td>
<td></td>
</tr>
<tr>
<td>Non-maskable interrupts (NMI), 249</td>
<td></td>
</tr>
<tr>
<td>Non-volatile random access memory (NVRAM), 55</td>
<td></td>
</tr>
<tr>
<td>Operating System, 9–10, 72, 268</td>
<td></td>
</tr>
<tr>
<td>Orthogonalization, 156</td>
<td></td>
</tr>
<tr>
<td>OS boot loader, 51</td>
<td></td>
</tr>
<tr>
<td>OS loader, 55</td>
<td></td>
</tr>
<tr>
<td>PCI, 49</td>
<td></td>
</tr>
<tr>
<td>PE/COFF, 64</td>
<td></td>
</tr>
<tr>
<td>PEI Modules (PEIM), 62</td>
<td></td>
</tr>
<tr>
<td>PEIM-to-PEIM Interface (PPI), 63</td>
<td></td>
</tr>
<tr>
<td>PERFiX, 255</td>
<td></td>
</tr>
<tr>
<td>Performance evaluation, 258</td>
<td></td>
</tr>
<tr>
<td>Periodic task, 18–21, 42, 251–252</td>
<td></td>
</tr>
<tr>
<td>Peripheral Component Interconnect (PCI), 64</td>
<td></td>
</tr>
<tr>
<td>Pl boot, 60</td>
<td></td>
</tr>
<tr>
<td>Platform-based design, 8</td>
<td></td>
</tr>
<tr>
<td>Platform Initialization (Pl), 59</td>
<td></td>
</tr>
<tr>
<td>Platform Management Interrupt (PMI), 66</td>
<td></td>
</tr>
<tr>
<td>Polling-based synchronization, 216</td>
<td></td>
</tr>
<tr>
<td>PORTOS, 254</td>
<td></td>
</tr>
<tr>
<td>POSIX, 240, 255</td>
<td></td>
</tr>
<tr>
<td>Power Management, 38</td>
<td></td>
</tr>
<tr>
<td>Power On Self Test (POST), 10, 48</td>
<td></td>
</tr>
<tr>
<td>PPort, 267</td>
<td></td>
</tr>
<tr>
<td>Pre-EFI Initialization (PEI), 60, 62</td>
<td></td>
</tr>
<tr>
<td>Preemptive scheduling, 72</td>
<td></td>
</tr>
<tr>
<td>Preemptive thread multitasking kernels, 38</td>
<td></td>
</tr>
<tr>
<td>Priority-based scheduler, 249</td>
<td></td>
</tr>
<tr>
<td>Processor core, 71, 87, 183, 243</td>
<td></td>
</tr>
<tr>
<td>Processor utilization factor, 22</td>
<td></td>
</tr>
<tr>
<td>Process technology, 7</td>
<td></td>
</tr>
<tr>
<td>Productivity, 4, 6</td>
<td></td>
</tr>
<tr>
<td>crash, 6</td>
<td></td>
</tr>
<tr>
<td>gap, 4</td>
<td></td>
</tr>
<tr>
<td>Programmers View (PV), 273</td>
<td></td>
</tr>
<tr>
<td>Project planning, 8</td>
<td></td>
</tr>
<tr>
<td>Pseudo-parallel, 242</td>
<td></td>
</tr>
<tr>
<td>Quality of Service (QoS), 40–42</td>
<td></td>
</tr>
<tr>
<td>Rate monotonic priority assignment, 21</td>
<td></td>
</tr>
<tr>
<td>Rate monotonic scheduling, 21, 241</td>
<td></td>
</tr>
<tr>
<td>Real mode, 50</td>
<td></td>
</tr>
<tr>
<td>Real-Time Operating System (RTOS), 16, 31, 220, 235, 283</td>
<td></td>
</tr>
<tr>
<td>Real-Time scheduling, 21, 72</td>
<td></td>
</tr>
<tr>
<td>Reconfiguration, 8</td>
<td></td>
</tr>
<tr>
<td>Reduced Instruction Set Computer (RISC), 3, 183</td>
<td></td>
</tr>
<tr>
<td>Register access class-based, 103</td>
<td></td>
</tr>
<tr>
<td>C struct, 104</td>
<td></td>
</tr>
<tr>
<td>function-based, 101</td>
<td></td>
</tr>
<tr>
<td>functions, 131</td>
<td></td>
</tr>
<tr>
<td>macro-based, 103</td>
<td></td>
</tr>
<tr>
<td>object-based, 101</td>
<td></td>
</tr>
<tr>
<td>structured approach, 115</td>
<td></td>
</tr>
<tr>
<td>Register addressable unit, 125</td>
<td></td>
</tr>
<tr>
<td>auto-shadow, 120</td>
<td></td>
</tr>
<tr>
<td>bit field structure, 129</td>
<td></td>
</tr>
<tr>
<td>block transfer, 121</td>
<td></td>
</tr>
<tr>
<td>file model, 159</td>
<td></td>
</tr>
<tr>
<td>indexing bit fields, 121</td>
<td></td>
</tr>
<tr>
<td>mirror size, 125</td>
<td></td>
</tr>
<tr>
<td>offset, 125</td>
<td></td>
</tr>
<tr>
<td>specification, 123</td>
<td></td>
</tr>
<tr>
<td>types, 166</td>
<td></td>
</tr>
<tr>
<td>width, 125</td>
<td></td>
</tr>
<tr>
<td>Regression, 163</td>
<td></td>
</tr>
<tr>
<td>Regression testing, 167</td>
<td></td>
</tr>
<tr>
<td>Response time, 258</td>
<td></td>
</tr>
<tr>
<td>Response time analysis, 23, 236</td>
<td></td>
</tr>
<tr>
<td>Reuse, 7</td>
<td></td>
</tr>
<tr>
<td>Round-Robin scheduling, 241, 249</td>
<td></td>
</tr>
<tr>
<td>RPort, 267</td>
<td></td>
</tr>
<tr>
<td>RTL, 160</td>
<td></td>
</tr>
<tr>
<td>RTOS</td>
<td></td>
</tr>
</tbody>
</table>
Abstraction Layer (RAL), 220
API, 254
case, 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
HARDWARE-DEPENDENT SOFTWARE
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