CONTENTS

Preface xxvii
Acknowledgments xxxiii

1 Overview of Embedded System 1

1.1 Introduction 1

1.1.1 Definition of an embedded system 1

1.1.2 Example systems 2

1.2 System design requirements 3

1.3 Embedded SoPC systems 4

1.3.1 Basic development flow 5

1.4 Book organization 8

1.5 Bibliographic notes 8

PART I  BASIC DIGITAL CIRCUITS DEVELOPMENT

2 Gate-level Combinational Circuit 11

2.1 Introduction 11

2.2 General description 12

2.3 Basic lexical elements and data types 13

2.3.1 Lexical elements 13

2.4 Data types 14
4.1.5 Concatenation and replication operators 57
4.1.6 Conditional operators 58
4.1.7 Operator precedence 59
4.1.8 Expression bit-length adjustment 59
4.1.9 Synthesis of $z$ and $x$ values 60
4.2 Always block for a combinational circuit 62
4.2.1 Basic syntax and behavior 62
4.2.2 Procedural assignment 63
4.2.3 Variable data types 63
4.2.4 Simple examples 64
4.3 If statement 65
4.3.1 Syntax 65
4.3.2 Examples 66
4.4 Case statement 68
4.4.1 Syntax 68
4.4.2 Examples 69
4.4.3 The casez and casex statements 69
4.4.4 Full case and parallel case 70
4.5 Routing structure of conditional control constructs 71
4.5.1 Priority routing network 71
4.5.2 Multiplexing network 73
4.6 General coding guidelines for an always block 74
4.6.1 Common errors in combinational circuit codes 74
4.6.2 Guidelines 77
4.7 Parameter and constant 78
4.7.1 Constant 78
4.7.2 Parameter 79
4.7.3 Use of parameters in Verilog-1995 81
4.8 Design examples 81
4.8.1 Hexadecimal digit to seven-segment LED decoder 81
4.8.2 Sign-magnitude adder 83
4.8.3 Barrel shifter 85
4.8.4 Simplified floating-point adder 87
4.9 Bibliographic notes 90
4.10 Suggested experiments 91
4.10.1 Multifunction barrel shifter 91
4.10.2 Dual-priority encoder 91
4.10.3 BCD incrementor 91
4.10.4 Floating-point greater-than circuit 92
4.10.5 Floating-point and signed integer conversion circuit 92
4.10.6 Enhanced floating-point adder 92
5 Regular Sequential Circuit

5.1 Introduction
  5.1.1 D FF and register
  5.1.2 Synchronous system
  5.1.3 Code development

5.2 HDL code of the FF and register
  5.2.1 D FF
  5.2.2 Register
  5.2.3 Register file
  5.2.4 SRAM

5.3 Simple design examples
  5.3.1 Shift register
  5.3.2 Binary counter and variant

5.4 Testbench for sequential circuits

5.5 Timing analysis
  5.5.1 Timing parameters
  5.5.2 Timing considerations in Quartus II

5.6 Case study
  5.6.1 Stopwatch
  5.6.2 FIFO buffer

5.7 Cyclone II device embedded memory module
  5.7.1 Overview of memory options of DE1 board
  5.7.2 Overview of embedded M4K module
  5.7.3 Methods to incorporate embedded memory module
  5.7.4 HDL module to infer synchronous single-port RAM
  5.7.5 HDL module to infer synchronous simple dual-port RAM
  5.7.6 HDL module to infer synchronous true dual-port RAM
  5.7.7 HDL module to infer synchronous ROM
  5.7.8 HDL module to specify RAM initial values
  5.7.9 FIFO buffer revisited

5.8 Bibliographic notes

5.9 Suggested experiments
  5.9.1 Programmable square-wave generator
  5.9.2 Pulse width modulation circuit
  5.9.3 Rotating square circuit
  5.9.4 Heartbeat circuit
  5.9.5 Rotating LED banner circuit
  5.9.6 Enhanced stopwatch
  5.9.7 FIFO with data width conversion
  5.9.8 Stack
  5.9.9 ROM-based sign-magnitude adder
  5.9.10 ROM-based temperature conversion
<table>
<thead>
<tr>
<th>6 FSM</th>
<th>137</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Introduction</td>
<td>137</td>
</tr>
<tr>
<td>6.1.1 Mealy and Moore outputs</td>
<td>138</td>
</tr>
<tr>
<td>6.1.2 FSM representation</td>
<td>138</td>
</tr>
<tr>
<td>6.2 FSM code development</td>
<td>140</td>
</tr>
<tr>
<td>6.3 Design examples</td>
<td>143</td>
</tr>
<tr>
<td>6.3.1 Rising-edge detector</td>
<td>143</td>
</tr>
<tr>
<td>6.3.2 Debouncing circuit</td>
<td>147</td>
</tr>
<tr>
<td>6.3.3 Testing circuit</td>
<td>151</td>
</tr>
<tr>
<td>6.4 Bibliographic notes</td>
<td>153</td>
</tr>
<tr>
<td>6.5 Suggested experiments</td>
<td>153</td>
</tr>
<tr>
<td>6.5.1 Dual-edge detector</td>
<td>153</td>
</tr>
<tr>
<td>6.5.2 Alternative debouncing circuit</td>
<td>153</td>
</tr>
<tr>
<td>6.5.3 Parking lot occupancy counter</td>
<td>153</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7 FSMD</th>
<th>155</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1 Introduction</td>
<td>155</td>
</tr>
<tr>
<td>7.1.1 Single RT operation</td>
<td>156</td>
</tr>
<tr>
<td>7.1.2 ASMD chart</td>
<td>156</td>
</tr>
<tr>
<td>7.1.3 Decision box with a register</td>
<td>158</td>
</tr>
<tr>
<td>7.2 Code development of an FSMD</td>
<td>161</td>
</tr>
<tr>
<td>7.2.1 Debouncing circuit based on RT methodology</td>
<td>161</td>
</tr>
<tr>
<td>7.2.2 Code with explicit data path components</td>
<td>161</td>
</tr>
<tr>
<td>7.2.3 Code with implicit data path components</td>
<td>164</td>
</tr>
<tr>
<td>7.2.4 Comparison</td>
<td>166</td>
</tr>
<tr>
<td>7.3 Design examples</td>
<td>168</td>
</tr>
<tr>
<td>7.3.1 Fibonacci number circuit</td>
<td>168</td>
</tr>
<tr>
<td>7.3.2 Division circuit</td>
<td>171</td>
</tr>
<tr>
<td>7.3.3 Binary-to-BCD conversion circuit</td>
<td>175</td>
</tr>
<tr>
<td>7.3.4 Period counter</td>
<td>178</td>
</tr>
<tr>
<td>7.3.5 Accurate low-frequency counter</td>
<td>181</td>
</tr>
<tr>
<td>7.4 Bibliographic notes</td>
<td>184</td>
</tr>
<tr>
<td>7.5 Suggested experiments</td>
<td>184</td>
</tr>
<tr>
<td>7.5.1 Alternative debouncing circuit</td>
<td>184</td>
</tr>
<tr>
<td>7.5.2 BCD-to-binary conversion circuit</td>
<td>184</td>
</tr>
<tr>
<td>7.5.3 Fibonacci circuit with BCD I/O: design approach 1</td>
<td>185</td>
</tr>
<tr>
<td>7.5.4 Fibonacci circuit with BCD I/O: design approach 2</td>
<td>185</td>
</tr>
<tr>
<td>7.5.5 Auto-scaled low-frequency counter</td>
<td>186</td>
</tr>
<tr>
<td>7.5.6 Reaction timer</td>
<td>186</td>
</tr>
<tr>
<td>7.5.7 Babbage difference engine emulation circuit</td>
<td>187</td>
</tr>
</tbody>
</table>
8 Selected Topics of Verilog

8.1 Blocking versus nonblocking assignment
8.1.1 Overview 189
8.1.2 Combinational circuit 191
8.1.3 Memory element 193
8.1.4 Sequential circuit with mixed blocking and nonblocking assignments 194

8.2 Alternative coding style for sequential circuit
8.2.1 Binary counter 196
8.2.2 FSM 198
8.2.3 FSMD 199
8.2.4 Summary 201

8.3 Use of the signed data type
8.3.1 Overview 201
8.3.2 Signed number in Verilog-1995 203
8.3.3 Signed number in Verilog-2001 203

8.4 Use of function in synthesis
8.4.1 Overview 204
8.4.2 Examples 205

8.5 Additional constructs for testbench development
8.5.1 Always block and initial block 207
8.5.2 Procedural statements 207
8.5.3 Timing control 209
8.5.4 Delay control 209
8.5.5 Event control 210
8.5.6 Wait statement 211
8.5.7 Timescale directive 211
8.5.8 System functions and tasks 212
8.5.9 User-defined functions and tasks 216
8.5.10 Example of a comprehensive testbench 217

8.6 Bibliographic notes 223

8.7 Suggested experiments
8.7.1 Shift register with blocking and nonblocking assignments 223
8.7.2 Alternative coding style for BCD counter 224
8.7.3 Alternative coding style for FIFO buffer 224
8.7.4 Alternative coding style for Fibonacci circuit 224
8.7.5 Dual-mode comparator 224
8.7.6 Enhanced binary counter monitor 224
8.7.7 Testbench for FIFO buffer 225

PART II BASIC NIOS II SOFTWARE DEVELOPMENT
## 9 Nios II Processor Overview

9.1 Introduction 229
9.2 Register file and ALU
   9.2.1 Register file 231
   9.2.2 ALU 231
9.3 Memory and I/O organization
   9.3.1 Nios II memory interface 232
   9.3.2 Overview of memory hierarchy 232
   9.3.3 Virtual memory 232
   9.3.4 Memory protection 233
   9.3.5 Cache memory 233
   9.3.6 Tightly coupled memory 234
   9.3.7 I/O organization 234
   9.3.8 Interconnect structure 235
9.4 Exception and interrupt handler 235
9.5 JTAG debug module 235
9.6 Bibliographic notes 235
9.7 Suggested projects
   9.7.1 Comparison of Nios II and MIPS 236

## 10 Nios II System Derivation and Low-Level Access

10.1 Development flow revisited 237
   10.1.1 Hardware development 237
   10.1.2 Software development 239
   10.1.3 Flashing-LED system 239
10.2 Nios II hardware generation tutorial
   10.2.1 Create a hardware project in Quartus II 240
   10.2.2 Create a Nios II system and generate HDL codes 240
   10.2.3 Create a top-level HDL file that instantiates the Nios II system 246
   10.2.4 Compiling and programming 247
10.3 Nios II SBT GUI tutorial 248
   10.3.1 Create BSP library 248
   10.3.2 Configure the BSP using BSP Editor 249
   10.3.3 Create user application directory and add application files 250
   10.3.4 Build and run software 251
   10.3.5 Check code size 252
10.4 System id core for hardware-software consistency 252
10.5 Direct low-level I/O access
   10.5.1 Review of C pointer 254
   10.5.2 C pointer for I/O register 255
10.6 Robust low-level I/O access 256
  10.6.1 system.h 256
  10.6.2 alt_types.h 257
  10.6.3 io.h 257
10.7 Some C techniques for low-level I/O operations 258
  10.7.1 Bit manipulation 258
  10.7.2 Packing and unpacking 258
10.8 Software development 259
  10.8.1 Basic embedded program architecture 259
  10.8.2 Main program and task routines 260
10.9 Bibliographic notes 261
10.10 Suggested experiments 261
  10.10.1 Chasing LED circuit 261
  10.10.2 Collision LED circuit 262
  10.10.3 Pulse width modulation circuit 262
  10.10.4 Rotating square circuit 262
  10.10.5 Heartbeat circuit 262
10.11 Complete program listing 263

11 Predesigned Nios II I/O Peripherals 265
11.1 Overviews 265
11.2 PIO core 266
  11.2.1 Configuration 266
  11.2.2 Register map 269
  11.2.3 Visible register 270
11.3 JTAG UART core 270
  11.3.1 Configuration 270
  11.3.2 Register map 271
11.4 Internal timer core 272
  11.4.1 Configuration 272
  11.4.2 Register map 273
11.5 Enhanced flashing-LED Nios II system 274
  11.5.1 SOPC design 274
  11.5.2 Top-level HDL file 278
11.6 Software development of enhanced flashing-LED system 279
  11.6.1 Introduction to device driver 280
  11.6.2 Program structure of the enhanced flashing-LED system 280
  11.6.3 Main program 281
  11.6.4 Function naming convention 281
11.7 Device driver routines 282
  11.7.1 Driver for PIO peripherals 282
  11.7.2 JTAG UART 284
## 11.7.3 Timer

11.8 Task routines

11.8.1 The flashsys_init_v1() function

11.8.2 The sw_get_command_v1() function

11.8.3 The jtaguart_disp_msg_v1() function

11.8.4 The sseg_disp_msg_v1() function

11.8.5 The led_flash_v1() function

11.9 Software construction and testing

11.10 Bibliographic notes

11.11 Suggested experiments

11.11.1 “Uptime” feature in flashing-LED system

11.11.2 Counting with different timer mode

11.11.3 JTAG UART input

11.11.4 Enhanced collision LED circuit

11.11.5 Rotating LED banner circuit

11.11.6 Enhanced stopwatch

11.11.7 Parking lot occupancy counter

11.11.8 Reaction timer with pushbutton switch control

11.11.9 Reaction timer with keyboard control

11.11.10 Communication with serial port

11.12 Complete program listing

## 12 Predesigned Nios II I/O Drivers and HAL API

12.1 Overview of HAL

12.1.1 Desktop-like and barebone embedded systems

12.1.2 HAL paradigm

12.1.3 Device classes

12.1.4 HAL-compliant device drivers

12.1.5 The _regs.h file

12.1.6 HAL-based initialization sequence

12.2 BSP

12.2.1 Overview

12.2.2 BSP file structure

12.2.3 BSP configuration

12.3 HAL-based flashing-LED program

12.3.1 Functions using generic I/O devices

12.3.2 Functions using non-generic I/O devices

12.3.3 Initialization routine and main program

12.3.4 Software construction and testing

12.4 Device driver consideration

12.4.1 I/O access methods

12.4.2 Comparisons
12.4.3 Device drivers in this book 320
12.5 Bibliographic notes 321
12.6 Suggested experiments 321
12.6.1 “Uptime” feature in flashing-LED system 321
12.6.2 Enhanced collision LED circuit 322
12.6.3 Parking lot occupancy counter 322
12.6.4 Reaction timer with keyboard control 322
12.6.5 Digital alarm clock 322
12.7 Complete program listing 323

13 Interrupt and ISR 325
13.1 Interrupt processing in the HAL framework 325
13.1.1 Overview 326
13.1.2 Interrupt controller of the Nios II processor 326
13.1.3 Top-level exception handler 327
13.1.4 Interrupt service routines 328
13.2 Interrupt-based flashing-LED program 328
13.2.1 Interrupt of timer core 329
13.2.2 Driver of timer core 329
13.2.3 ISR version 1 330
13.2.4 ISR version 2 332
13.3 Interrupt and scheduling 333
13.3.1 Scheduling 333
13.3.2 Performance 335
13.4 Bibliographic notes 336
13.5 Suggested experiments 336
13.5.1 Flashing-LED system with pushbutton switch ISR 336
13.5.2 ISR-driven flashing-LED system 336
13.5.3 “Uptime” feature in flashing-LED system 337
13.5.4 Reaction timer with keyboard control 337
13.5.5 Digital alarm clock 337
13.6 Complete program listing 338

PART III CUSTOM I/O PERIPHERAL DEVELOPMENT

14 Custom I/O Peripheral with PIO Cores 345
14.1 Introduction 345
14.2 Integration of division circuit to a Nios II system 346
14.2.1 PIO modules 346
14.2.2 Integration 347
14.3 Testing 347
14.4 Suggested experiments 350
14.4.1 Division core ISR
14.4.2 Division core with eight-bit data
14.4.3 Division core with 64-bit data
14.4.4 Fibonacci number circuit
14.4.5 Period counter

15 Avalon Interconnect and SOPC Component
15.1 Introduction
15.2 Avalon MM interface
15.2.1 Avalon MM slave interface signals
15.2.2 Avalon MM slave interface properties
15.2.3 Avalon MM slave timing
15.3 System interconnect fabric for Avalon interface
15.4 SOPC I/O component wrapping circuit
15.4.1 Interface I/O buffer
15.4.2 Memory alignment
15.4.3 Output decoding from an Avalon MM master
15.4.4 Input multiplexing to an Avalon MM master
15.4.5 Practical consideration
15.5 SOPC component construction tutorial
15.5.1 Avalon interfaces
15.5.2 Register map
15.5.3 Wrapped division circuit
15.5.4 SOPC component creation
15.5.5 SOPC component instantiation
15.6 Testing
15.7 Bibliographic notes
15.8 Suggested experiments
15.8.1 Division core ISR
15.8.2 Alternative buffering scheme for the division core
15.8.3 Division core with eight-bit data
15.8.4 Division core with 64-bit data
15.8.5 Fibonacci number circuit
15.8.6 Period counter

16 SRAM and SDRAM Controllers
16.1 Memory resources of DE1 board
16.2 Brief overview of timing and clock management
16.2.1 Clock distribution network
16.2.2 Timing consideration of off-chip access
16.2.3 PLL
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.3</td>
<td>Overview of SRAM</td>
<td>389</td>
</tr>
<tr>
<td>16.3.1</td>
<td>SRAM cell</td>
<td>389</td>
</tr>
<tr>
<td>16.3.2</td>
<td>Basic organization</td>
<td>390</td>
</tr>
<tr>
<td>16.3.3</td>
<td>Timing</td>
<td>391</td>
</tr>
<tr>
<td>16.3.4</td>
<td>IS61LV25616AL SRAM device</td>
<td>393</td>
</tr>
<tr>
<td>16.4</td>
<td>SRAM controller IP core</td>
<td>394</td>
</tr>
<tr>
<td>16.4.1</td>
<td>Avalon interfaces</td>
<td>394</td>
</tr>
<tr>
<td>16.4.2</td>
<td>Controller circuit</td>
<td>396</td>
</tr>
<tr>
<td>16.4.3</td>
<td>SOPC component creation</td>
<td>397</td>
</tr>
<tr>
<td>16.5</td>
<td>Overview of DRAM</td>
<td>398</td>
</tr>
<tr>
<td>16.5.1</td>
<td>DRAM cell</td>
<td>398</td>
</tr>
<tr>
<td>16.5.2</td>
<td>Basic DRAM organization</td>
<td>400</td>
</tr>
<tr>
<td>16.5.3</td>
<td>DRAM timing</td>
<td>401</td>
</tr>
<tr>
<td>16.6</td>
<td>Overview of SDRAM</td>
<td>403</td>
</tr>
<tr>
<td>16.6.1</td>
<td>Basic SDRAM organization</td>
<td>403</td>
</tr>
<tr>
<td>16.6.2</td>
<td>SDRAM timing</td>
<td>404</td>
</tr>
<tr>
<td>16.6.3</td>
<td>ICSI IS42S16400 SDRAM device</td>
<td>406</td>
</tr>
<tr>
<td>16.7</td>
<td>SDRAM controller and PLL</td>
<td>406</td>
</tr>
<tr>
<td>16.7.1</td>
<td>Basic SDRAM controller</td>
<td>406</td>
</tr>
<tr>
<td>16.7.2</td>
<td>SDRAM controller IP core</td>
<td>408</td>
</tr>
<tr>
<td>16.7.3</td>
<td>SOPC PLL IP core</td>
<td>408</td>
</tr>
<tr>
<td>16.8</td>
<td>Testing system</td>
<td>411</td>
</tr>
<tr>
<td>16.8.1</td>
<td>Testing hardware configuration</td>
<td>411</td>
</tr>
<tr>
<td>16.8.2</td>
<td>Testing software</td>
<td>415</td>
</tr>
<tr>
<td>16.9</td>
<td>Bibliographic notes</td>
<td>418</td>
</tr>
<tr>
<td>16.10</td>
<td>Suggested experiments</td>
<td>418</td>
</tr>
<tr>
<td>16.10.1</td>
<td>SRAM controller without I/O register</td>
<td>418</td>
</tr>
<tr>
<td>16.10.2</td>
<td>SRAM controller speed test</td>
<td>418</td>
</tr>
<tr>
<td>16.10.3</td>
<td>SRAM controller with Avalon MM tristate interface</td>
<td>419</td>
</tr>
<tr>
<td>16.10.4</td>
<td>SDRAM controller clock skew test</td>
<td>419</td>
</tr>
<tr>
<td>16.10.5</td>
<td>Memory performance comparison</td>
<td>419</td>
</tr>
<tr>
<td>16.10.6</td>
<td>Effect of cache memory</td>
<td>419</td>
</tr>
<tr>
<td>16.10.7</td>
<td>SDRAM controller from scratch</td>
<td>419</td>
</tr>
<tr>
<td>16.11</td>
<td>Complete program listing</td>
<td>420</td>
</tr>
</tbody>
</table>

17 PS2 Keyboard and Mouse  | 423 |

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.1</td>
<td>Introduction</td>
<td>423</td>
</tr>
<tr>
<td>17.2</td>
<td>PS2 receiving subsystem</td>
<td>424</td>
</tr>
<tr>
<td>17.2.1</td>
<td>PS2-device-to-host communication protocol</td>
<td>424</td>
</tr>
<tr>
<td>17.2.2</td>
<td>Design and code</td>
<td>425</td>
</tr>
<tr>
<td>17.3</td>
<td>PS2 transmitting subsystem</td>
<td>428</td>
</tr>
<tr>
<td>17.3.1</td>
<td>Host-to-PS2-device communication protocol</td>
<td>428</td>
</tr>
</tbody>
</table>
CONTENTS

17.3.2 Design and code
17.4 Complete PS2 system
17.5 PS2 controller IP core development
  17.5.1 Avalon interfaces
  17.5.2 Register map
  17.5.3 Wrapped PS2 system
  17.5.4 SOPC component creation
17.6 PS2 driver
  17.6.1 Register map
  17.6.2 Write routines
  17.6.3 Read routines
17.7 Keyboard driver
  17.7.1 Overview of the scan code
  17.7.2 Interaction with host
  17.7.3 Driver routines
17.8 Mouse driver
  17.8.1 Overview of PS2 mouse protocol
  17.8.2 Interaction with host
  17.8.3 Driver routines
17.9 Test
17.10 Use of book’s custom IP cores
  17.10.1 File organization
  17.10.2 SOPC library integration
  17.10.3 Comprehensive Nios II testing system
17.11 Bibliographic notes
17.12 Suggested experiments
  17.12.1 PS2 receiving subsystem with watchdog timer
  17.12.2 Software receiving FIFO
  17.12.3 Software PS2 controller
  17.12.4 Keyboard-controlled LED flashing circuit
  17.12.5 Enhanced keyboard driver routine I
  17.12.6 Enhanced keyboard driver routine II
  17.12.7 Remote-mode mouse driver
  17.12.8 Scroll-wheel mouse driver
17.13 Complete program listing

18 VGA Controller

18.1 Introduction
  18.1.1 Basic operation of a CRT
  18.1.2 VGA port of the DE1 board
  18.1.3 Video controller
18.2 VGA synchronization
18.2.1 Horizontal synchronization 480
18.2.2 Vertical synchronization 481
18.2.3 Timing calculation of VGA synchronization signals 482
18.2.4 HDL implementation 482
18.3 SRAM-based video RAM controller 484
  18.3.1 Overview of video memory 484
  18.3.2 Memory consideration of DE1 board 485
  18.3.3 Ad hoc SRAM controller 486
  18.3.4 HDL code 491
18.4 Palette circuit 494
18.5 Video controller IP core development 495
  18.5.1 Complete video controller 495
  18.5.2 Avalon interfaces 495
  18.5.3 Register map 496
  18.5.4 Wrapped video controller 496
  18.5.5 SOPC component creation 497
18.6 Video driver 498
  18.6.1 Video memory access routines 498
  18.6.2 Geometrical model routine 499
  18.6.3 Bitmap processing routines 500
  18.6.4 Bit-mapped text routines 503
18.7 Mouse processing routines 506
18.8 Testing program 507
  18.8.1 Chart plotting routine 509
  18.8.2 General plotting functions 510
  18.8.3 Strip swapping routine 512
  18.8.4 Mouse demonstration routine 512
  18.8.5 Bit-mapped text routine 513
18.9 Bitmap file processing 514
  18.9.1 BMP format overview 514
  18.9.2 Generation of BMP file 515
  18.9.3 Sprite-based design 515
  18.9.4 BMP file access 516
  18.9.5 Host-based file system 517
  18.9.6 Bitmap file retrieval routines 519
18.10 Bibliographic notes 522
18.11 Suggested experiments 523
  18.11.1 PLL-based VGA controller 523
  18.11.2 VGA controller with 16-bit memory configuration 523
  18.11.3 VGA controller with 3-bit color depth 523
  18.11.4 VGA controller with 1-bit color depth 523
  18.11.5 VGA controller with double buffering 523
18.11.6 VGA controller with 320-by-240 resolution 523
18.11.7 VGA controller with vertical mode operation 524
18.11.8 Geometrical model functions 524
18.11.9 Bitmap manipulation functions 524
18.11.10 Simulated “Etch A Sketch” toy 524
18.11.11 Palette lookup table circuit 524
18.11.12 Virtual LED flashing system panel 525
18.11.13 Virtual analog wall clock 525

18.12 Suggested projects 525
18.12.1 Configurable VGA controller 525
18.12.2 VGA controller using system SDRAM 525
18.12.3 Paint program 525
18.12.4 Video game 526

18.13 Complete program listing 527

19 Audio Codec Controller 555
19.1 Introduction 555
19.1.1 Overview of codec 555
19.1.2 Overview of WM8731 device 556
19.1.3 Registers of WM8731 device 557

19.2 I²C controller 560
19.2.1 Overview of I²C interface 560
19.2.2 HDL implementation 562

19.3 Codec data access controller 568
19.3.1 Overview of digital audio interface 568
19.3.2 HDL implementation 569

19.4 Audio codec controller IP core development 572
19.4.1 Complete audio codec controller 572
19.4.2 Avalon interfaces 574
19.4.3 Register map 575
19.4.4 Wrapped audio codec controller 575
19.4.5 SOPC component creation 577

19.5 Codec driver 577
19.5.1 I²C command routines 577
19.5.2 Data source select routine 578
19.5.3 Device initialization routine 578
19.5.4 Audio data access routines 579

19.6 Testing program 580

19.7 Audio file processing 583
19.7.1 WAV format overview 583
19.7.2 Audio format conversion program 584
19.7.3 Audio data retrieval routine 585
xxii CONTENTS

19.8 Bibliographic notes 587
19.9 Suggested experiments 587
  19.9.1 Software I2C controller 587
  19.9.2 Hardware data access controller using master clocking mode 587
  19.9.3 Software data access controller using slave clocking mode 587
  19.9.4 Software data access controller using master clocking mode 587
  19.9.5 Configurable data access controller 588
  19.9.6 Voice recorder 588
  19.9.7 Real-time sinusoidal wave generator 588
  19.9.8 Real-time audio wave display 588
  19.9.9 Echo effect 588
19.10 Suggested projects 589
  19.10.1 Full-fledged I2C controller 589
  19.10.2 Digital equalizer 589
  19.10.3 Digital audio oscilloscope 589
19.11 Complete program listing 590

20 SD Card Controller 601
  20.1 Overview of SD card 601
  20.2 SPI controller 602
    20.2.1 Overview of SPI interface 602
    20.2.2 HDL implementation 603
  20.3 SPI controller IP core development 606
    20.3.1 Avalon interfaces 606
    20.3.2 Register map 606
    20.3.3 Wrapped SPI controller 607
    20.3.4 SOPC component creation 608
  20.4 SD card protocol 608
    20.4.1 SD card command and response formats 608
    20.4.2 Initialization and identification process 610
    20.4.3 Data read and write process 611
  20.5 SPI and SD card driver 613
    20.5.1 SPI driver routines 613
    20.5.2 SD card driver routines 614
  20.6 File access 619
    20.6.1 Overview of FAT16 structure 620
    20.6.2 Read-only FAT16 file access driver routines 625
  20.7 Testing program 632
  20.8 Performance of SD card data transfer 636
  20.9 Bibliographic notes 637
  20.10 Suggested experiments 637
    20.10.1 SD card data transfer performance test 637
## CONTENTS

20.10.2 Robust SD card driver routines 637  
20.10.3 Dedicated processor for SD card access 638  
20.10.4 Hardware-based SD card read and write operation 638  
20.10.5 SD card information retrieval 638  
20.10.6 MMC card support 638  
20.10.7 Multiple sector read and write operation 638  
20.10.8 SD card driver routines with CRC checking 639  
20.10.9 Digital music player 639  
20.10.10 Digital picture frame 639  
20.10.11 Additional FAT functionalities 639

20.11 Suggested projects 639  
20.11.1 HAL API file access integration 639

20.12 Complete program listing 640

## PART IV HARDWARE ACCELERATOR CASE STUDIES

### 21 GCD Accelerator 663

21.1 Introduction 663  
21.2 Software implementation 664  
21.3 Hardware implementation 665  
21.3.1 ASMD chart 665  
21.3.2 HDL implementation 665

21.4 Time measurement 668  
21.4.1 HAL time stamp driver 668  
21.4.2 Custom hardware counter 669

21.5 GCD accelerator IP core development 669  
21.5.1 Avalon interfaces 669  
21.5.2 Register map 669  
21.5.3 Wrapped GCD accelerator 669

21.6 Testing program 671  
21.6.1 GCD routines 671  
21.6.2 Main program 673

21.7 Performance comparison 673  
21.8 Bibliographic notes 674

21.9 Suggested experiments 675  
21.9.1 Performance with other processor configuration 675  
21.9.2 GCD accelerator with minimal size 675  
21.9.3 GCD accelerator with trailing zero circuit 675  
21.9.4 GCD accelerator with 64-bit data 675  
21.9.5 GCD accelerator with 128-bit data 675  
21.9.6 GCD by Euclid’s algorithm 675

21.10 Complete program listing 676
22 Mandelbrot Set Fractal Accelerator

22.1 Introduction

22.1.1 Overview of the Mandelbrot set
22.1.2 Determination of a Mandelbrot set point
22.1.3 Coloring scheme
22.1.4 Generation of a fractal image

22.2 Fixed-point arithmetic

22.3 Software implementation of calc_frac_point()

22.4 Hardware implementation of calc_frac_point()

22.4.1 ASMD chart
22.4.2 HDL implementation

22.5 Mandelbrot set fractal accelerator IP core development

22.5.1 Avalon interface
22.5.2 Register map
22.5.3 Wrapped Mandelbrot set fractal accelerator

22.6 Testing program

22.6.1 Fractal graphic user interface
22.6.2 Fractal hardware accelerator engine control routine
22.6.3 Fractal drawing routine
22.6.4 Text panel display routines
22.6.5 Mouse processing routine
22.6.6 Main program

22.7 Discussion

22.8 Bibliographic notes

22.9 Suggested experiments

22.9.1 Hardware accelerator with one multiplier
22.9.2 Hardware accelerator with modified escape condition
22.9.3 Hardware accelerator with Q4.12 format
22.9.4 Hardware accelerator with multiple fractal engines
22.9.5 “Burning-ship” fractal
22.9.6 Enhanced testing program

22.10 Suggested projects

22.10.1 Floating-point hardware accelerator
22.10.2 General fractal drawing platform

22.11 Complete program listing

23 Direct Digital Frequency Synthesis

23.1 Introduction

23.2 Design and implementation

23.2.1 Direct synthesis of a digital waveform
23.2.2 Direct synthesis of an unmodulated analog waveform
23.2.3 Direct synthesis of a modulated analog waveform 718
23.2.4 HDL implementation 718
23.3 DDFS IP core development 721
  23.3.1 Avalon interface 721
  23.3.2 Register map 721
  23.3.3 Wrapped DDFS circuit 721
  23.3.4 Codec DAC integration 723
23.4 DDFS driver 723
  23.4.1 Configuration routines 724
  23.4.2 Initialization routine 724
23.5 Testing 725
  23.5.1 Overview of music notes and synthesis 725
  23.5.2 Testing program 726
23.6 Bibliographic notes 730
23.7 Suggested experiments 730
  23.7.1 Quadrature phase carrier generation 730
  23.7.2 Reduced-size phase-to-amplitude lookup table 731
  23.7.3 Synthetic music player 731
  23.7.4 Keyboard piano 731
  23.7.5 Keyboard recorder 731
  23.7.6 Hardware envelope generator 731
  23.7.7 Additive harmonic synthesis 731
  23.7.8 Sample-based synthesis 732
23.8 Suggested projects 732
  23.8.1 Sound generator 732
  23.8.2 Function generator 732
  23.8.3 Full-fledged electric synthesizer 732
23.9 Complete program listing 733

References 741

Topic Index 745