CONTENTS

Preface xxvii
Acknowledgments xxxiii

PART I BASIC DIGITAL CIRCUITS DEVELOPMENT

1 Gate-Level Combinational Circuit 1

1.1 Introduction 1
1.1.1 Brief history of Verilog and SystemVerilog 1
1.1.2 Book coverage 2
1.2 General description 3
1.3 Basic lexical elements and data types 4
1.3.1 Lexical elements 4
1.3.2 Data types used in the book 5
1.3.3 Number representation 6
1.3.4 Operators 7
1.4 Program skeleton 7
1.4.1 Port declaration 7
1.4.2 Signal declaration 8
1.4.3 Program body 8
1.4.4 Concurrent semantics 9
1.4.5 Another example 10
1.5 Structural description 10
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6</td>
<td>Top-level signal mapping</td>
<td>13</td>
</tr>
<tr>
<td>1.7</td>
<td>Testbench</td>
<td>14</td>
</tr>
<tr>
<td>1.8</td>
<td>Bibliographic notes</td>
<td>16</td>
</tr>
<tr>
<td>1.9</td>
<td>Suggested experiments</td>
<td>16</td>
</tr>
<tr>
<td>1.9.1</td>
<td>Code for gate-level greater-than circuit</td>
<td>17</td>
</tr>
<tr>
<td>1.9.2</td>
<td>Code for gate-level binary decoder</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>Overview of FPGA and EDA Software</td>
<td>19</td>
</tr>
<tr>
<td>2.1</td>
<td>FPGA</td>
<td>19</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Overview of a general FPGA device</td>
<td>19</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Overview of the Xilinx Artix-7 devices</td>
<td>20</td>
</tr>
<tr>
<td>2.2</td>
<td>Overview of the Digilent Nexys 4 DDR board</td>
<td>21</td>
</tr>
<tr>
<td>2.3</td>
<td>Development flow</td>
<td>22</td>
</tr>
<tr>
<td>2.4</td>
<td>Xilinx Vivado Design Suite</td>
<td>24</td>
</tr>
<tr>
<td>2.5</td>
<td>Bibliographic notes</td>
<td>24</td>
</tr>
<tr>
<td>2.6</td>
<td>Suggested experiments</td>
<td>24</td>
</tr>
<tr>
<td>2.6.1</td>
<td>Gate-level greater-than circuit</td>
<td>24</td>
</tr>
<tr>
<td>2.6.2</td>
<td>Gate-level binary decoder</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>RT-Level Combinational Circuit</td>
<td>29</td>
</tr>
<tr>
<td>3.1</td>
<td>Operators</td>
<td>29</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Arithmetic operators</td>
<td>31</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Shift operators</td>
<td>31</td>
</tr>
<tr>
<td>3.1.3</td>
<td>Relational and equality operators</td>
<td>32</td>
</tr>
<tr>
<td>3.1.4</td>
<td>Bitwise, reduction, and logical operators</td>
<td>32</td>
</tr>
<tr>
<td>3.1.5</td>
<td>Concatenation and replication operators</td>
<td>33</td>
</tr>
<tr>
<td>3.1.6</td>
<td>Conditional operators</td>
<td>34</td>
</tr>
<tr>
<td>3.1.7</td>
<td>Operator precedence</td>
<td>35</td>
</tr>
<tr>
<td>3.1.8</td>
<td>Expression bit-length adjustment</td>
<td>35</td>
</tr>
<tr>
<td>3.1.9</td>
<td>Synthesis of z and x values</td>
<td>36</td>
</tr>
<tr>
<td>3.2</td>
<td>Always block for a combinational circuit</td>
<td>38</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Overview of always block</td>
<td>39</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Procedural assignment</td>
<td>40</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Conceptual examples</td>
<td>40</td>
</tr>
<tr>
<td>3.3</td>
<td>Coding guidelines</td>
<td>43</td>
</tr>
<tr>
<td>3.4</td>
<td>If statement</td>
<td>43</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Syntax</td>
<td>43</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Examples</td>
<td>44</td>
</tr>
<tr>
<td>3.5</td>
<td>Case statement</td>
<td>45</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Syntax</td>
<td>45</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Examples</td>
<td>46</td>
</tr>
</tbody>
</table>
3.5.3 The casez and casex statements 47
3.5.4 Full case and parallel case 48
3.6 Routing structure of conditional control constructs 49
  3.6.1 Priority routing network 49
  3.6.2 Multiplexing network 51
3.7 Additional coding guidelines for an always block 52
  3.7.1 Common errors in combinational circuit codes 52
  3.7.2 Guidelines 56
3.8 Parameter and constant 56
  3.8.1 Constant 56
  3.8.2 Parameter 58
3.9 Replicated structure 59
  3.9.1 Generate-for statement 59
  3.9.2 Procedural-for statement 60
  3.9.3 Example 60
3.10 Design examples 62
  3.10.1 Hexadecimal digit to seven-segment LED decoder 62
  3.10.2 Sign-magnitude adder 65
  3.10.3 Barrel shifter 68
  3.10.4 Simplified floating-point adder 69
3.11 Bibliographic notes 73
3.12 Suggested experiments 73
  3.12.1 Multi-function barrel shifter 73
  3.12.2 Parameterized barrel shifter 74
  3.12.3 Dual-priority encoder 74
  3.12.4 BCD incrementor 74
  3.12.5 Floating-point greater-than circuit 74
  3.12.6 Floating-point and signed integer conversion circuit 74
  3.12.7 Enhanced floating-point adder 75

4 Regular Sequential Circuit 77
4.1 Introduction 77
  4.1.1 D FF and register 78
  4.1.2 Basic block system 78
  4.1.3 Code development 79
  4.1.4 Sequential circuit coding guidelines and style 79
4.2 HDL code of the FF and register 80
  4.2.1 D FF 80
  4.2.2 Register 85
4.3 Simple design examples 85
  4.3.1 Shift register 85
  4.3.2 Binary counter and variant 87
6.2 Code development of an FSMD
   6.2.1 Debouncing circuit based on RT methodology 137
   6.2.2 Code with explicit data path components 137
   6.2.3 Code with implicit data path components 140
   6.2.4 Comparison 142

6.3 Design examples
   6.3.1 Fibonacci number circuit 144
   6.3.2 Division circuit 147
   6.3.3 Binary-to-BCD conversion circuit 150
   6.3.4 Period counter 153
   6.3.5 Accurate low-frequency counter 156

6.4 Bibliographic notes 159
6.5 Suggested experiments
   6.5.1 Early detection debouncing circuit 159
   6.5.2 BCD-to-binary conversion circuit 160
   6.5.3 Fibonacci circuit with BCD I/O: design approach 1 160
   6.5.4 Fibonacci circuit with BCD I/O: design approach 2 160
   6.5.5 Auto-scaled low-frequency counter 161
   6.5.6 Reaction timer 161
   6.5.7 Babbage difference engine emulation circuit 162

7 RAM and Buffer of FPGA

7.1 Embedded memory of FPGA device 165
   7.1.1 Memory of an Artix device 166
   7.1.2 Memory available in the Nexys 4 DDR board 166
7.2 General description for a RAM-like component 167
   7.2.1 Register file 167
   7.2.2 Dynamic array indexing operation 169
   7.2.3 Key aspects of a RAM module 170
   7.2.4 Genuine ROM 171
7.3 FIFO buffer 173
   7.3.1 FIFO read configuration 174
   7.3.2 Circular queue implementation 175
7.4 HDL templates for memory inference 178
   7.4.1 Methods to incorporate memory modules 178
   7.4.2 Synchronous dual-port RAM 179
   7.4.3 “Simple” synchronous dual-port RAM 180
   7.4.4 Synchronous single-port RAM 181
   7.4.5 Synchronous ROM 182
   7.4.6 BRAM-based FIFO buffer 183
   7.4.7 Design considerations 183
7.5 Overview of memory controller 184
7.6 Bibliographic notes 185
7.7 Suggested experiments 186
7.7.1 ROM-based sign-magnitude adder 186
7.7.2 ROM-based temperature conversion 186
7.7.3 FIFO with data width conversion 186
7.7.4 Standard FIFO to FWFT FIFO conversion circuit 187
7.7.5 FIFO buffer with extended status 187
7.7.6 Stack 187

8 Selected Topics of SystemVerilog 189
8.1 Timing model 189
8.1.1 Concurrent constructs 190
8.1.2 Assignment statement 190
8.1.3 Basic model 190
8.1.4 Blocking versus nonblocking assignment 192
8.2 Coding guidelines revisited 194
8.2.1 “Single variable assignment” guideline 195
8.2.2 “Blocking assignment for combinational circuit” guideline 195
8.2.3 “Nonblocking assignment for register” guideline 197
8.3 Alternative coding style 198
8.3.1 First coding style revisited 198
8.3.2 Sequential circuit with mixed blocking and nonblocking assignments 199
8.3.3 Combined coding style 201
8.3.4 Summary 206
8.4 Data types 206
8.4.1 The net and variable types 206
8.4.2 The logic data type 207
8.4.3 Limitation of the logic data type 208
8.4.4 New data types in SystemVerilog 208
8.5 Use of the signed data type 209
8.5.1 Overview 209
8.5.2 Signed number conversion 210
8.6 Bibliographic notes 211
8.7 Suggested experiments 211
8.7.1 Shift register with blocking and nonblocking assignments 211
8.7.2 Alternative coding style for the BCD counter 212
8.7.3 Alternative coding style for the FIFO buffer 212
8.7.4 Alternative coding style for the Fibonacci circuit 212
8.7.5 Dual-mode comparator 212
## CONTENTS

### PART II EMBEDDED SOC I: VANILLA FPRO SYSTEM

### 9 Overview of Embedded SoC Systems

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1 Embedded SoC</td>
<td>215</td>
</tr>
<tr>
<td>9.1.1 Overview of embedded systems</td>
<td>215</td>
</tr>
<tr>
<td>9.1.2 FPGA-based SoC</td>
<td>216</td>
</tr>
<tr>
<td>9.1.3 IP cores</td>
<td>216</td>
</tr>
<tr>
<td>9.2 Development flow of the embedded SoC</td>
<td>217</td>
</tr>
<tr>
<td>9.2.1 Hardware–software partition</td>
<td>217</td>
</tr>
<tr>
<td>9.2.2 Hardware development flow</td>
<td>217</td>
</tr>
<tr>
<td>9.2.3 Software development flow</td>
<td>219</td>
</tr>
<tr>
<td>9.2.4 Physical implementation and test</td>
<td>219</td>
</tr>
<tr>
<td>9.2.5 Custom IP core development</td>
<td>219</td>
</tr>
<tr>
<td>9.3 FPro SoC Platform</td>
<td>220</td>
</tr>
<tr>
<td>9.3.1 Motivations</td>
<td>220</td>
</tr>
<tr>
<td>9.3.2 Platform hardware organization</td>
<td>221</td>
</tr>
<tr>
<td>9.3.3 Platform software organization</td>
<td>223</td>
</tr>
<tr>
<td>9.3.4 Modified development flow</td>
<td>224</td>
</tr>
<tr>
<td>9.4 Adaptation on the Digilent Nexys 4 DDR board</td>
<td>224</td>
</tr>
<tr>
<td>9.5 Portability</td>
<td>226</td>
</tr>
<tr>
<td>9.5.1 Processor Module and Bridge</td>
<td>226</td>
</tr>
<tr>
<td>9.5.2 MMIO subsystem</td>
<td>227</td>
</tr>
<tr>
<td>9.5.3 Video subsystem</td>
<td>227</td>
</tr>
<tr>
<td>9.6 Organization</td>
<td>228</td>
</tr>
<tr>
<td>9.7 Bibliographic notes</td>
<td>228</td>
</tr>
</tbody>
</table>

### 10 Bare Metal System Software Development

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1 Bare metal system development overview</td>
<td>231</td>
</tr>
<tr>
<td>10.1.1 Desktop-like system versus bare metal system</td>
<td>231</td>
</tr>
<tr>
<td>10.1.2 Basic embedded program architecture</td>
<td>232</td>
</tr>
<tr>
<td>10.2 Memory-mapped I/O</td>
<td>233</td>
</tr>
<tr>
<td>10.2.1 Overview</td>
<td>233</td>
</tr>
<tr>
<td>10.2.2 Memory alignment</td>
<td>234</td>
</tr>
<tr>
<td>10.2.3 I/O register map</td>
<td>234</td>
</tr>
<tr>
<td>10.2.4 I/O address space of the FPro system</td>
<td>234</td>
</tr>
<tr>
<td>10.3 Direct I/O Register Access</td>
<td>235</td>
</tr>
<tr>
<td>10.3.1 Review of C pointer</td>
<td>235</td>
</tr>
<tr>
<td>10.3.2 C pointer for I/O register</td>
<td>236</td>
</tr>
<tr>
<td>10.4 Robust I/O register access</td>
<td>237</td>
</tr>
<tr>
<td>10.4.1 chu_io_map.h and chu_io_map.svh</td>
<td>237</td>
</tr>
<tr>
<td>10.4.2 inttypes.h</td>
<td>238</td>
</tr>
</tbody>
</table>
CONTENTS

10.4.3 chu_io_rw.h

10.5 Techniques for low-level I/O operations
10.5.1 Bit manipulation
10.5.2 Packing and unpacking

10.6 Device Drivers
10.6.1 Overview
10.6.2 GPO and GPI drivers
10.6.3 Timer driver
10.6.4 UART driver

10.7 FPro utility routines and directory structure
10.7.1 Minimal hardware requirements
10.7.2 Utility routines
10.7.3 Directory structure

10.8 Test program
10.8.1 IP core verification routine
10.8.2 Programming with limited memory
10.8.3 Test function integration
10.8.4 Test program for the vanilla FPro system
10.8.5 Implementation

10.9 Bibliographic notes

10.10 Suggested experiments
10.10.1 Chasing LEDs
10.10.2 Collision LEDs
10.10.3 Pulse width modulation
10.10.4 System time display

11 FPro Bus Protocol and MMIO Slot Specification

11.1 FPro bus
11.1.1 Overview of the bus
11.1.2 SoC interconnect
11.1.3 FPro bus protocol specification

11.2 Interface with the bus
11.2.1 Introduction
11.2.2 Write interface and decoding
11.2.3 Read interface and multiplexing
11.2.4 FIFO buffer as an I/O register
11.2.5 Timing consideration

11.3 MMIO I/O core
11.3.1 MMIO slot interface specification
11.3.2 Basic MMIO I/O core construction
11.3.3 GPO and GPI cores

11.4 Timer core development
11.4.1 Custom logic 270
11.4.2 Register map 271
11.4.3 Wrapping circuit for the slot interface 271
11.5 MMIO controller 272
  11.5.1 chu_io_map.svh file 273
  11.5.2 HDL code 273
  11.5.3 Vanilla MMIO subsystem 275
11.6 MCS I/O bus and bridge 278
  11.6.1 Overview of Xilinx MicroBlaze MCS 278
  11.6.2 MicroBlaze MCS I/O bus 278
  11.6.3 MCS-to-FPro bridge 279
11.7 Vanilla FPro system construction 281
11.8 Bibliographic notes 282
11.9 Suggested experiments 283
  11.9.1 FPro bus with a byte-lane enable signal 283
  11.9.2 Seven-segment control with a GPO core 283
  11.9.3 GPIO core 283
  11.9.4 Blinking-LED core 284
  11.9.5 Timer core with a programmable period 284
  11.9.6 Timer core with a run-once mode 284

12 UART Core 287
12.1 Introduction 287
  12.1.1 Overview of serial communication 287
  12.1.2 Overview of the UART 288
  12.1.3 Oversampling procedure 288
12.2 UART construction 289
  12.2.1 Conceptual design 289
  12.2.2 Baud rate generator 290
  12.2.3 UART receiver 291
  12.2.4 UART transmitter 293
  12.2.5 Top-level HDL code 295
12.3 UART core development 296
  12.3.1 Register map 296
  12.3.2 Wrapping circuit for the slot interface 297
12.4 UART driver 298
  12.4.1 Class definition 299
  12.4.2 Basic methods 300
  12.4.3 ASCII code 301
  12.4.4 Display methods 303
  12.4.5 Test 305
12.5 Additional project ideas 305
12.5.1 Original serial port 305
12.5.2 Emulated serial port 305
12.5.3 Direct connection 306
12.5.4 USB-to-UART adaptor 306
12.5.5 Wireless adaptor 307
12.6 Bibliographic notes 308
12.7 Suggested experiments 308
  12.7.1 UART-controlled chasing LEDs 308
  12.7.2 Alternative read configuration 308
  12.7.3 UART controller with a parity bit 308
  12.7.4 UART core with an error status 309
  12.7.5 Configurable UART core 309
  12.7.6 UART core with automatic baud rate detection 309
  12.7.7 UART core with enhanced automatic baud rate detection 310
  12.7.8 UART core with an automatic baud rate and a parity detection circuit 310

PART III EMBEDDED SOC II:
BASIC I/O CORES

13 Xilinx XADC Core 313
  13.1 Overview of XADC 313
    13.1.1 Block diagram 313
    13.1.2 Configuration 314
  13.2 XADC core development 315
    13.2.1 XADC instantiation 315
    13.2.2 Basic wrapping circuit design 316
    13.2.3 Register map 318
    13.2.4 HDL code 318
  13.3 XADC core device driver 320
    13.3.1 Class definition 320
    13.3.2 Class implementation 321
    13.3.3 Testing for the XADC core 322
  13.4 Sampler FPPro system 323
    13.4.1 Testing procedure of an FPPro core 323
    13.4.2 System configuration 323
    13.4.3 Hardware derivation 324
    13.4.4 Software verification program 331
  13.5 Additional project ideas 332
  13.6 Bibliographic notes 333
  13.7 Suggested experiments 333
    13.7.1 Real-time voltage display 333
13.7.2 Potentiometer-controlled chasing LEDs 333  
13.7.3 Potentiometer-controlled LED dimmer 333  
13.7.4 Enhanced wrapping circuit: part I 333  
13.7.5 Enhanced wrapping circuit: part II 333

14 Pulse Width Modulation Core 335  
14.1 Introduction 335  
14.1.1 PWM as analog output 335  
14.1.2 Main characteristics 336  
14.2 PWM design 336  
14.2.1 Basic design 336  
14.2.2 Enhanced design 337  
14.3 PWM core development 339  
14.3.1 Register map 339  
14.3.2 Wrapped PWM circuit 340  
14.4 PWM driver 341  
14.4.1 Class definition 341  
14.4.2 Class implementation 342  
14.5 Testing 343  
14.6 Project ideas 343  
14.7 Suggested experiments 345  
14.7.1 Police dash light 345  
14.7.2 Rainbow night light 345  
14.7.3 Enhanced PWM core: part I 345  
14.7.4 Enhanced PWM core: part II 346  
14.7.5 Enhanced GPIO core 346  
14.7.6 Servo motor driver 346

15 Debouncing Core and LED-Mux Core 347  
15.1 Debouncing Core 347  
15.1.1 Multi-bit debouncing circuit 347  
15.1.2 Register map and the slot wrapping circuit 350  
15.1.3 Driver 351  
15.1.4 Test 352  
15.2 LED-mux core 352  
15.2.1 Eight-digit seven-segment LED display multiplexing circuit 352  
15.2.2 Register map and the slot wrapping circuit 354  
15.2.3 Driver 355  
15.2.4 Test 358  
15.3 Project ideas 358  
15.4 Suggested experiments 360
CONTENTS

15.4.1 Area comparison of two debouncing circuits 360
15.4.2 Enhanced debouncing core: part I 360
15.4.3 Enhanced debouncing core: part II 360
15.4.4 Rotating square pattern revisited 360
15.4.5 Heartbeat pattern revisited 360
15.4.6 Stopwatch 360
15.4.7 Enhanced LED-mux core 361

16 SPI Core 363

16.1 Overview 363
16.1.1 Conceptual architecture 364
16.1.2 Multiple-device configuration 364
16.1.3 Basic timing 366
16.1.4 Operation modes 367
16.1.5 Undefined aspects 368

16.2 SPI controller 369
16.2.1 Basic design 369
16.2.2 FSMD construction 370
16.2.3 HDL implementation 370

16.3 SPI core development 374
16.3.1 Register map 374
16.3.2 Wrapping circuit for the slot interface 374

16.4 SPI driver 376
16.4.1 Class definition 376
16.4.2 Class implementation 377

16.5 Test 378
16.5.1 ADXL362 accelerometer 378
16.5.2 Test program 380

16.6 Project ideas 381
16.6.1 SD card 381
16.6.2 TFT LCD module 382

16.7 Bibliographic notes 382

16.8 Suggested experiments 382
16.8.1 Inclination sensing 382
16.8.2 “Tapping” detection 382
16.8.3 ADXL362 C++ class 383
16.8.4 Enhanced SPI controller: part I 383
16.8.5 Enhanced SPI controller: part II 383
16.8.6 “Automatic-read” ADXL362 wrapper: part I 383
16.8.7 “Automatic-read” ADXL362 wrapper: part II 384
16.8.8 Flash memory access 384
16.8.9 SPI slave controller: part I 384
16.8.10 SPI slave controller: part II 385

17 I²C Core 387

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.1</td>
<td>Overview</td>
<td>387</td>
</tr>
<tr>
<td>17.1.1</td>
<td>Electrical characteristics</td>
<td>388</td>
</tr>
<tr>
<td>17.1.2</td>
<td>Basic bus protocol</td>
<td>388</td>
</tr>
<tr>
<td>17.1.3</td>
<td>Basic timing</td>
<td>389</td>
</tr>
<tr>
<td>17.1.4</td>
<td>Additional features</td>
<td>390</td>
</tr>
<tr>
<td>17.2</td>
<td>I²C controller</td>
<td>391</td>
</tr>
<tr>
<td>17.2.1</td>
<td>Basic design</td>
<td>391</td>
</tr>
<tr>
<td>17.2.2</td>
<td>Conceptual FSMD construction</td>
<td>391</td>
</tr>
<tr>
<td>17.2.3</td>
<td>Output control logic</td>
<td>394</td>
</tr>
<tr>
<td>17.2.4</td>
<td>I²C bus clock generation</td>
<td>394</td>
</tr>
<tr>
<td>17.2.5</td>
<td>HDL implementation</td>
<td>395</td>
</tr>
<tr>
<td>17.3</td>
<td>I²C core development</td>
<td>400</td>
</tr>
<tr>
<td>17.3.1</td>
<td>Register map</td>
<td>400</td>
</tr>
<tr>
<td>17.3.2</td>
<td>Wrapping circuit for the slot interface</td>
<td>400</td>
</tr>
<tr>
<td>17.4</td>
<td>I²C driver</td>
<td>401</td>
</tr>
<tr>
<td>17.4.1</td>
<td>Class definition</td>
<td>401</td>
</tr>
<tr>
<td>17.4.2</td>
<td>Class implementation</td>
<td>402</td>
</tr>
<tr>
<td>17.5</td>
<td>Test</td>
<td>405</td>
</tr>
<tr>
<td>17.5.1</td>
<td>ADT7420 temperature sensor</td>
<td>405</td>
</tr>
<tr>
<td>17.5.2</td>
<td>Test program</td>
<td>406</td>
</tr>
<tr>
<td>17.6</td>
<td>Project idea</td>
<td>406</td>
</tr>
<tr>
<td>17.7</td>
<td>Bibliographic notes</td>
<td>407</td>
</tr>
<tr>
<td>17.8</td>
<td>Suggested experiments</td>
<td>407</td>
</tr>
<tr>
<td>17.8.1</td>
<td>Thermometer</td>
<td>407</td>
</tr>
<tr>
<td>17.8.2</td>
<td>ADT7420 C++ class</td>
<td>407</td>
</tr>
<tr>
<td>17.8.3</td>
<td>Enhanced I²C core</td>
<td>408</td>
</tr>
<tr>
<td>17.8.4</td>
<td>“Automatic-read” ADT7420 wrapper</td>
<td>408</td>
</tr>
<tr>
<td>17.8.5</td>
<td>I²C slave controller: part I</td>
<td>408</td>
</tr>
<tr>
<td>17.8.6</td>
<td>I²C slave controller: part II</td>
<td>408</td>
</tr>
</tbody>
</table>

18 PS2 Core 409

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.1</td>
<td>Introduction</td>
<td>409</td>
</tr>
<tr>
<td>18.1.1</td>
<td>PS2-device-to-host communication protocol and timing</td>
<td>410</td>
</tr>
<tr>
<td>18.1.2</td>
<td>Host-to-PS2-device communication protocol and timing</td>
<td>410</td>
</tr>
<tr>
<td>18.2</td>
<td>PS2 controller</td>
<td>411</td>
</tr>
<tr>
<td>18.2.1</td>
<td>Conceptual design</td>
<td>411</td>
</tr>
<tr>
<td>18.2.2</td>
<td>PS2 receiving subsystem</td>
<td>411</td>
</tr>
<tr>
<td>18.2.3</td>
<td>PS2 transmitting subsystem</td>
<td>415</td>
</tr>
</tbody>
</table>
CONTENTS

18.2.4 Complete PS2 system 419
18.3 PS2 core development 420
  18.3.1 Register map 420
  18.3.2 Wrapping circuit for the slot interface 421
18.4 PS2 driver 422
  18.4.1 Class definition 422
  18.4.2 Lower layer methods 422
  18.4.3 PS2 initialization routine 423
  18.4.4 Keyboard routine 425
  18.4.5 Mouse routine 428
18.5 Test 430
18.6 Bibliographic notes 431
18.7 Suggested experiments 431
  18.7.1 PS2 receiving subsystem with watchdog timer 431
  18.7.2 Keyboard-controlled LED flashing circuit 432
  18.7.3 Enhanced keyboard driver routine: part I 432
  18.7.4 Enhanced keyboard driver routine: part II 432
  18.7.5 Remote-mode mouse driver 432
  18.7.6 Scroll-wheel mouse driver 432

19 Sound I: DDFS Core 433
19.1 Introduction 433
19.2 Design and implementation 434
  19.2.1 Direct synthesis of a digital waveform 434
  19.2.2 Direct synthesis of an unmodulated analog waveform 435
  19.2.3 Direct synthesis of a modulated analog waveform 436
19.3 Fixed-point arithmetic 437
19.4 DDFS construction 438
19.5 DAC (digital-to-analog converter) 440
  19.5.1 Conceptual design 440
  19.5.2 HDL implementation 441
19.6 DDFS core development 442
  19.6.1 Register map 442
  19.6.2 Wrapping circuit for the slot interface 443
19.7 DDFS driver 444
  19.7.1 Class definition 444
  19.7.2 Class implementation 445
19.8 Test 447
19.9 Bibliographic notes 448
19.10 Suggested experiments 448
  19.10.1 Quadrature phase carrier generation 448
  19.10.2 Reduced-size phase-to-amplitude lookup table 448
CONTENTS

19.10.3 Additive harmonic synthesis 449
19.10.4 Simple function generator 449
19.10.5 Arbitrary waveform generator 449
19.10.6 Sample-based synthesis 449

20.1 Introduction 451
20.2 ADSR envelope generator 452
  20.2.1 Conceptual FSMD design 453
  20.2.2 ASMD chart 453
  20.2.3 HDL implementation 455
20.3 ADSR core development 457
  20.3.1 Register map 457
  20.3.2 Wrapped ADSR circuit 458
20.4 ADSR driver 460
  20.4.1 Class definition 460
  20.4.2 Configuration methods 461
  20.4.3 calc_note_freq() method 463
  20.4.4 play_note() method 465
20.5 Test 465
20.6 Project idea 466
20.7 Bibliographic notes 467
20.8 Suggested experiments 467
  20.8.1 RTTTL music player 467
  20.8.2 ADSR envelope testing 467
  20.8.3 Pushbutton piano 467
  20.8.4 Keyboard piano 468
  20.8.5 Keyboard recorder 468
  20.8.6 Real-time mode ADSR generator 468
  20.8.7 Real-time mode pushbutton piano 468
  20.8.8 Merged DDFS and ADSR core 468
  20.8.9 ADSR core with an automatic play FIFO buffer 468
  20.8.10 ADSR core for frequency modulation 468

PART IV EMBEDDED SOC III: VIDEO CORES

21.1 Introduction to a video display 471
  21.1.1 Conceptual video display 471
  21.1.2 VGA interface 472
21.2 Stream interface 473
CONTENTS

21.2.1 Random-access interface versus stream interface 473
21.2.2 Flow control of the stream interface 473
21.3 VGA synchronization 475
  21.3.1 Basic operation of a CRT monitor 475
  21.3.2 Horizontal synchronization 476
  21.3.3 Vertical synchronization 478
  21.3.4 Pixel clock rate 479
  21.3.5 VGA synchronization circuit 480
21.4 Bar test-pattern generator 483
21.5 Color-to-grayscale conversion circuit 485
21.6 Demo video system 486
21.7 Advanced video standards 488
21.8 Bibliographic notes 489
21.9 Suggested experiments 489
  21.9.1 Horizontal bar test-pattern generator 489
  21.9.2 Color channel selection circuit 489
  21.9.3 Enhanced color-to-grayscale conversion circuit 489
  21.9.4 Square test-pattern generator: part I 489
  21.9.5 Square test-pattern generator: part II 489
  21.9.6 Square test-pattern generator: part III 490
  21.9.7 Square test-pattern generator: part IV 490

22 FPro Video Subsystem 491

  22.1 Organization of the video subsystem 491
    22.1.1 Overview 491
    22.1.2 Video controller 493
    22.1.3 HDL of the video controller 494
  22.2 FPro video IP core 495
    22.2.1 Basic functionality 495
    22.2.2 Blending operation 496
    22.2.3 Core architecture 498
    22.2.4 Alternative core partition 500
  22.3 Example video cores 500
    22.3.1 Bar test-pattern generator core 500
    22.3.2 Color-to-grayscale conversion core 503
    22.3.3 “Dummy” core 504
  22.4 FPro video synchronization core 504
    22.4.1 Line buffer 505
    22.4.2 Enhanced video synchronization circuit 508
    22.4.3 HDL code 511
  22.5 Daisy video subsystem 512
    22.5.1 Subsystem overview 512
22.5.2 Interface to the video synchronization core 513
22.5.3 HDL code 513
22.5.4 Timing and performance considerations 517

22.6 Vanilla daisy FPro system 517
22.6.1 Clock management core 518
22.6.2 Updated chu_io_map.svh 519
22.6.3 HDL code 519

22.7 Video driver and test program 521
22.7.1 Updated chu_io_map.h and chu_io_rw.h files 521
22.7.2 GPV core driver 522
22.7.3 Test program 523

22.8 Bibliographic notes 524

22.9 Suggested experiments 525
22.9.1 Color channel selection core 525
22.9.2 Enhanced color-to-grayscale conversion core 525
22.9.3 Square test-pattern generator core 525
22.9.4 Alpha blending circuit 525
22.9.5 “Highlight” core 525
22.9.6 SVGA synchronization core 526
22.9.7 Configurable video synchronization core 526
22.9.8 Pipelined video subsystem 526

23 Sprite Core 527

23.1 Introduction 527
23.2 Basic design 528
23.2.1 Sprite RAM 528
23.2.2 In-region comparison circuit 529
23.3 Mouse pointer core 530
23.3.1 Pointer sprite RAM 530
23.3.2 Pixel generation circuit 531
23.3.3 Top-level design 532
23.4 “Ghost” character core 534
23.4.1 Multiple images and animation 534
23.4.2 Overview of the palette scheme 535
23.4.3 Ghost sprite RAM and the palette circuit 535
23.4.4 Animation timing circuit 537
23.4.5 Pixel generation circuit 537
23.4.6 Top-level design 540
23.5 Sprite core driver and test program 541
23.5.1 Sprite core driver 541
23.5.2 Test program 543
23.6 Bibliographic notes 544
23.7  Suggested experiments 544
  23.7.1 Mouse pointer control with PS2 core 544
  23.7.2 Emulated ghost core 544
  23.7.3 Palette circuit for the mouse pointer sprite 544
  23.7.4 Sprite scaling circuit 544
  23.7.5 Portrait mode display 545
  23.7.6 Multiple-object generation 545
  23.7.7 Animation speed control 545
  23.7.8 Imitated blinking LED: part I 545
  23.7.9 Imitated blinking LED: part II 545
  23.7.10 Imitated blinking LED: part III 546

24 On-Screen-Display Core 547
  24.1 Introduction to tile graphics 547
  24.2 Basic OSD design 549
    24.2.1 Text-mode display 549
    24.2.2 Font ROM 550
    24.2.3 Tile RAM 550
    24.2.4 Basic organization 551
  24.3 OSD core 552
    24.3.1 Font ROM 552
    24.3.2 Pixel generation circuit 553
    24.3.3 Top-level design 555
  24.4 OSD core driver and test program 557
    24.4.1 OSD core driver 557
    24.4.2 Testing program 558
  24.5 Bibliographic notes 559
  24.6 Suggested experiments 559
    24.6.1 Rotating banner 559
    24.6.2 Text console 559
    24.6.3 Underline for the cursor 559
    24.6.4 Portrait-mode display 560
    24.6.5 Font scaling circuit: part I 560
    24.6.6 Font scaling circuit: part II 560
    24.6.7 Extended font 560
    24.6.8 Tile-based ghost core 560

25 VGA Frame Buffer Core 561
  25.1 Overview 561
  25.2 Frame buffer core 562
    25.2.1 FPGA memory consideration 562
25.2.2 Video memory module 562
25.2.3 Address translation 563
25.2.4 Pixel generation circuit 564
25.2.5 Register map 566
25.2.6 Top-level HDL code 566
25.3 Driver and test program 567
   25.3.1 Frame buffer core driver 567
   25.3.2 Geometrical modeling 568
   25.3.3 Test program 570
25.4 Project ideas 570
25.5 Bibliographic notes 572
25.6 Suggested experiments 572
   25.6.1 Virtual prototyping board panel 572
   25.6.2 Virtual analog wall clock 572
   25.6.3 Geometrical model functions 572
   25.6.4 Simulated “Etch a Sketch” toy 572
   25.6.5 Frame buffer core with 3-bit color depth 573
   25.6.6 Frame buffer core with 1-bit color depth 573
   25.6.7 QVGA frame buffer core 573
   25.6.8 Line drawing hardware accelerator 573
   25.6.9 Bidirectional frame buffer access: part I 573
   25.6.10 Bidirectional frame buffer access: part II 573

PART V EPILOGUE

26 What’s Next 577

References 581

Appendix A: Tutorials 585
   A.1 Overview of Xilinx Vivado IDE 585
   A.2 Short tutorial on Vivado hardware development 589
      A.2.1 Create a design project 590
      A.2.2 Add or create Xilinx IP core instances 591
      A.2.3 Add or create HDL design files 591
      A.2.4 Add a constraint file 592
      A.2.5 Perform synthesis, implementation, and bitstream generation 593
      A.2.6 Program an FPGA device 593
   A.3 Short tutorial on Vivado simulation 594
      A.3.1 Add or create HDL testbench 596
      A.3.2 Perform initial simulation 596
      A.3.3 Customize waveform display 597
## CONTENTS

**A.4 Tutorial on IP instantiation** 597
  - A.4.1 Dual-clock FIFO core via HDL templates 598
  - A.4.2 IP Catalog utility 599
  - A.4.3 Generate a MicroBlaze MCS component 600
  - A.4.4 XADC IP core 601
  - A.4.5 Clock management IP core 602

**A.5 Short tutorial on FPro system development** 604
  - A.5.1 Derive FPro system hardware 605
  - A.5.2 Export hardware configuration 605
  - A.5.3 Derive software 605
  - A.5.4 Embed elf file into FPGA’s memory module and regenerate bitstream 608
  - A.5.5 Set up the terminal emulator program 610
  - A.5.6 Program an FPGA device 610

**A.6 Bibliographic notes** 611

**Topic Index** 613