

Appendix A: List of Worked Examples

| | |
|---|-----|
| EXAMPLE 2.1 DECLARING VERILOG MODULE PORTS | 19 |
| EXAMPLE 3.1 BEHAVIORAL MODEL OF A 4-BIT ADDER IN VERILOG | 28 |
| EXAMPLE 3.2 COMBINATIONAL LOGIC USING CONTINUOUS ASSIGNMENT WITH LOGICAL OPERATORS | 30 |
| EXAMPLE 3.3 3-TO-8 ONE-HOT DECODER—VERILOG MODELING USING LOGICAL OPERATORS | 31 |
| EXAMPLE 3.4 7-SEGMENT DISPLAY DECODER—TRUTH TABLE | 32 |
| EXAMPLE 3.5 7-SEGMENT DISPLAY DECODER—LOGIC SYNTHESIS BY HAND | 33 |
| EXAMPLE 3.6 7-SEGMENT DISPLAY DECODER—VERILOG MODELING USING LOGICAL OPERATORS | 34 |
| EXAMPLE 3.7 4-TO-2 BINARY ENCODER—LOGIC SYNTHESIS BY HAND | 35 |
| EXAMPLE 3.8 4-TO-2 BINARY ENCODER—VERILOG MODELING USING LOGICAL OPERATORS | 35 |
| EXAMPLE 3.9 4-TO-1 MULTIPLEXER—VERILOG MODELING USING LOGICAL OPERATORS | 36 |
| EXAMPLE 3.10 1-TO-4 DEMULTIPLEXER—VERILOG MODELING USING LOGICAL OPERATORS | 37 |
| EXAMPLE 3.11 COMBINATIONAL LOGIC USING CONTINUOUS ASSIGNMENT WITH CONDITIONAL OPERATORS (1) | 38 |
| EXAMPLE 3.12 COMBINATIONAL LOGIC USING CONTINUOUS ASSIGNMENT WITH CONDITIONAL OPERATORS (2) | 39 |
| EXAMPLE 3.13 3-TO-8 ONE-HOT DECODER—VERILOG MODELING USING CONDITIONAL OPERATORS | 39 |
| EXAMPLE 3.14 7-SEGMENT DISPLAY DECODER—VERILOG MODELING USING CONDITIONAL OPERATORS | 40 |
| EXAMPLE 3.15 4-TO-2 BINARY ENCODER—VERILOG MODELING USING CONDITIONAL OPERATORS | 41 |
| EXAMPLE 3.16 4-TO-1 MULTIPLEXER—VERILOG MODELING USING CONDITIONAL OPERATORS | 41 |
| EXAMPLE 3.17 1-TO-4 DEMULTIPLEXER—VERILOG MODELING USING CONDITIONAL OPERATORS | 42 |
| EXAMPLE 3.18 MODELING DELAY IN CONTINUOUS ASSIGNMENTS | 44 |
| EXAMPLE 3.19 INERTIAL DELAY MODELING WHEN USING CONTINUOUS ASSIGNMENT | 45 |
| EXAMPLE 4.1 VERILOG STRUCTURAL DESIGN USING EXPLICIT PORT MAPPING | 52 |
| EXAMPLE 4.2 VERILOG STRUCTURAL DESIGN USING POSITIONAL PORT MAPPING | 53 |
| EXAMPLE 4.3 MODELING COMBINATIONAL LOGIC CIRCUITS USING GATE-LEVEL PRIMITIVES | 54 |
| EXAMPLE 4.4 MODELING COMBINATIONAL LOGIC CIRCUITS WITH A USER-DEFINED PRIMITIVE | 55 |
| EXAMPLE 4.5 DESIGN OF A HALF ADDER | 56 |
| EXAMPLE 4.6 DESIGN OF A FULL ADDER | 57 |
| EXAMPLE 4.7 DESIGN OF A FULL ADDER OUT OF HALF ADDERS | 58 |
| EXAMPLE 4.8 DESIGN OF A 4-BIT RIPPLE CARRY ADDER (RCA) | 59 |
| EXAMPLE 4.9 STRUCTURAL MODEL OF A FULL ADDER USING TWO HALF ADDERS | 60 |
| EXAMPLE 4.10 STRUCTURAL MODEL OF A 4-BIT RIPPLE CARRY ADDER IN VERILOG | 61 |
| EXAMPLE 5.1 USING BLOCKING ASSIGNMENTS TO MODEL COMBINATIONAL LOGIC | 69 |
| EXAMPLE 5.2 USING NON-BLOCKING ASSIGNMENTS TO MODEL SEQUENTIAL LOGIC | 69 |
| EXAMPLE 5.3 IDENTICAL BEHAVIOR WHEN USING BLOCKING VS. NON-BLOCKING ASSIGNMENTS | 70 |
| EXAMPLE 5.4 DIFFERENT BEHAVIOR WHEN USING BLOCKING VS. NON-BLOCKING ASSIGNMENTS (1) | 71 |
| EXAMPLE 5.5 DIFFERENT BEHAVIOR WHEN USING BLOCKING VS. NON-BLOCKING ASSIGNMENTS (2) | 72 |
| EXAMPLE 5.6 BEHAVIOR OF STATEMENT GROUPS BEGIN/END VS. FORK/JOIN | 73 |
| EXAMPLE 5.7 USING IF-ELSE STATEMENTS TO MODEL COMBINATIONAL LOGIC | 75 |
| EXAMPLE 5.8 USING CASE STATEMENTS TO MODEL COMBINATIONAL LOGIC | 76 |
| EXAMPLE 6.1 TEST BENCH FOR A COMBINATIONAL LOGIC CIRCUIT WITH MANUAL STIMULUS GENERATION | 90 |
| EXAMPLE 6.2 TEST BENCH FOR A SEQUENTIAL LOGIC CIRCUIT | 91 |
| EXAMPLE 6.3 PRINTING TEST BENCH RESULTS TO THE TRANSCRIPT | 92 |
| EXAMPLE 6.4 USING A LOOP TO GENERATE STIMULUS IN A TEST BENCH | 94 |
| EXAMPLE 6.5 TEST BENCH WITH AUTOMATIC OUTPUT CHECKING | 95 |
| EXAMPLE 6.6 PRINTING TEST BENCH RESULTS TO AN EXTERNAL FILE | 97 |
| EXAMPLE 6.7 READING TEST BENCH STIMULUS VECTORS FROM AN EXTERNAL FILE | 98 |
| EXAMPLE 7.1 BEHAVIORAL MODEL OF A D-LATCH IN VERILOG | 103 |
| EXAMPLE 7.2 BEHAVIORAL MODEL OF A D-FLIP-FLOP IN VERILOG | 104 |
| EXAMPLE 7.3 BEHAVIORAL MODEL OF A D-FLIP-FLOP WITH ASYNCHRONOUS RESET IN VERILOG | 105 |
| EXAMPLE 7.4 BEHAVIORAL MODEL OF A D-FLIP-FLOP WITH ASYNCHRONOUS RESET AND PRESET IN VERILOG | 106 |
| EXAMPLE 7.5 BEHAVIORAL MODEL OF A D-FLIP-FLOP WITH SYNCHRONOUS ENABLE IN VERILOG | 107 |
| EXAMPLE 7.6 RTL MODEL OF AN 8-BIT REGISTER IN VERILOG | 108 |

| | |
|--|-----|
| EXAMPLE 7.7 RTL MODEL OF A 4-STAGE, 8-BIT SHIFT REGISTER IN VERILOG | 109 |
| EXAMPLE 7.8 REGISTERS AS AGENTS ON A DATA BUS—SYSTEM TOPOLOGY | 110 |
| EXAMPLE 7.9 REGISTERS AS AGENTS ON A DATA BUS—RTL MODEL IN VERILOG | 110 |
| EXAMPLE 7.10 REGISTERS AS AGENTS ON A DATA BUS—SIMULATION WAVEFORM | 111 |
| EXAMPLE 8.1 PUSH-BUTTON WINDOW CONTROLLER IN VERILOG—DESIGN DESCRIPTION | 114 |
| EXAMPLE 8.2 PUSH-BUTTON WINDOW CONTROLLER IN VERILOG—PORT DEFINITION | 114 |
| EXAMPLE 8.3 PUSH-BUTTON WINDOW CONTROLLER IN VERILOG—FULL MODEL | 117 |
| EXAMPLE 8.4 PUSH-BUTTON WINDOW CONTROLLER IN VERILOG—SIMULATION WAVEFORM | 118 |
| EXAMPLE 8.5 PUSH-BUTTON WINDOW CONTROLLER IN VERILOG—CHANGING STATE CODES | 118 |
| EXAMPLE 8.6 SERIAL BIT SEQUENCE DETECTOR IN VERILOG—DESIGN DESCRIPTION AND PORT DEFINITION | 119 |
| EXAMPLE 8.7 SERIAL BIT SEQUENCE DETECTOR IN VERILOG—FULL MODEL | 120 |
| EXAMPLE 8.8 SERIAL BIT SEQUENCE DETECTOR IN VERILOG—SIMULATION WAVEFORM | 121 |
| EXAMPLE 8.9 VENDING MACHINE CONTROLLER IN VERILOG—DESIGN DESCRIPTION AND PORT DEFINITION | 121 |
| EXAMPLE 8.10 VENDING MACHINE CONTROLLER IN VERILOG—FULL MODEL | 122 |
| EXAMPLE 8.11 VENDING MACHINE CONTROLLER IN VERILOG—SIMULATION WAVEFORM | 123 |
| EXAMPLE 8.12 2-BIT UP/DOWN COUNTER IN VERILOG—DESIGN DESCRIPTION AND PORT DEFINITION | 123 |
| EXAMPLE 8.13 2-BIT UP/DOWN COUNTER IN VERILOG—FULL MODEL (THREE-BLOCK APPROACH) | 124 |
| EXAMPLE 8.14 2-BIT UP/DOWN COUNTER IN VERILOG—SIMULATION WAVEFORM | 124 |
| EXAMPLE 9.1 BINARY COUNTER USING A SINGLE PROCEDURAL BLOCK IN VERILOG | 129 |
| EXAMPLE 9.2 BINARY COUNTER WITH RANGE CHECKING IN VERILOG | 130 |
| EXAMPLE 9.3 BINARY COUNTER WITH ENABLE IN VERILOG | 131 |
| EXAMPLE 9.4 BINARY COUNTER WITH LOAD IN VERILOG | 132 |
| EXAMPLE 10.1 BEHAVIORAL MODELS OF A 4×4 ASYNCHRONOUS READ-ONLY MEMORY IN VERILOG | 137 |
| EXAMPLE 10.2 BEHAVIORAL MODELS OF A 4×4 SYNCHRONOUS READ-ONLY MEMORY IN VERILOG | 138 |
| EXAMPLE 10.3 BEHAVIORAL MODEL OF A 4×4 ASYNCHRONOUS READ/WRITE MEMORY IN VERILOG | 139 |
| EXAMPLE 10.4 BEHAVIORAL MODEL OF A 4×4 SYNCHRONOUS READ/WRITE MEMORY IN VERILOG | 140 |
| EXAMPLE 11.1 MEMORY MAP FOR A 256×8 MEMORY SYSTEM | 148 |
| EXAMPLE 11.2 EXECUTION OF AN INSTRUCTION TO “LOAD REGISTER A USING IMMEDIATE ADDRESSING” | 151 |
| EXAMPLE 11.3 EXECUTION OF AN INSTRUCTION TO “LOAD REGISTER A USING DIRECT ADDRESSING” | 152 |
| EXAMPLE 11.4 EXECUTION OF AN INSTRUCTION TO “STORE REGISTER A USING DIRECT ADDRESSING” | 153 |
| EXAMPLE 11.5 EXECUTION OF AN INSTRUCTION TO “ADD REGISTERS A AND B” | 154 |
| EXAMPLE 11.6 EXECUTION OF AN INSTRUCTION TO “BRANCH ALWAYS” | 155 |
| EXAMPLE 11.7 EXECUTION OF AN INSTRUCTION TO “BRANCH IF EQUAL TO ZERO” | 156 |
| EXAMPLE 11.8 TOP-LEVEL BLOCK DIAGRAM FOR THE 8-BIT COMPUTER SYSTEM | 158 |
| EXAMPLE 11.9 INSTRUCTION SET FOR THE 8-BIT COMPUTER SYSTEM | 159 |
| EXAMPLE 11.10 MEMORY SYSTEM BLOCK DIAGRAM FOR THE 8-BIT COMPUTER SYSTEM | 160 |
| EXAMPLE 11.11 CPU BLOCK DIAGRAM FOR THE 8-BIT COMPUTER SYSTEM | 164 |
| EXAMPLE 11.12 STATE DIAGRAM FOR LDA_IMM | 171 |
| EXAMPLE 11.13 SIMULATION WAVEFORM FOR LDA_IMM | 172 |
| EXAMPLE 11.14 STATE DIAGRAM FOR LDA_DIR | 173 |
| EXAMPLE 11.15 SIMULATION WAVEFORM FOR LDA_DIR | 174 |
| EXAMPLE 11.16 STATE DIAGRAM FOR STA_DIR | 175 |
| EXAMPLE 11.17 SIMULATION WAVEFORM FOR STA_DIR | 176 |
| EXAMPLE 11.18 STATE DIAGRAM FOR ADD_AB | 177 |
| EXAMPLE 11.19 SIMULATION WAVEFORM FOR ADD_AB | 178 |
| EXAMPLE 11.20 STATE DIAGRAM FOR BRA | 179 |
| EXAMPLE 11.21 SIMULATION WAVEFORM FOR BRA | 180 |
| EXAMPLE 11.22 STATE DIAGRAM FOR BEQ | 181 |
| EXAMPLE 11.23 SIMULATION WAVEFORM FOR BEQ WHEN TAKING THE BRANCH ($Z = 1$) | 182 |
| EXAMPLE 11.24 SIMULATION WAVEFORM FOR BEQ WHEN THE BRANCH IS NOT TAKEN ($Z = 0$) | 183 |

Index

A

Abstraction, 4

C

Capacity, 135, 136

Classical digital design flow, 8

Computer system design, 143

 addressing modes, 149

 arithmetic logic unit (ALU), 145, 146

 central processing unit, 144, 145

 condition code register, 145

 control unit, 145

 data memory, 144

 data path, 145

 direct addressing, 149

 example 8-bit system, 157, 158

 CPU, 163, 164

 detailed instruction execution, 170, 171

 instruction set, 158, 159

 memory system, 159, 160

 general-purpose registers, 145

 hardware, 143

 immediate addressing, 149

 inherent addressing, 150

 input output ports, 144

 instruction register, 145

 instructions, 143

 branches, 154, 155

 data manipulations, 153

 loads and stores, 150, 151

 memory address register, 145

 memory map, 147

 memory mapped system, 146

 opcodes, 149

 operands, 149

 program, 143

 program counter, 145

 program memory, 144

 registers, 145

 software, 143, 148

Counters, 131

 modeling in Verilog, 131

D

Design abstraction, 4

Design domains, 5

 behavioral domain, 5

 physical domain, 5

 structural domain, 5

Design levels, 5

 algorithmic level, 5

 circuit level, 5

 gate-level, 5

 register transfer level, 5

 system level, 5

Digital design flow, 8

F

Full adders, 56, 57

G

Gajski and Kuhn's Y-chart, 5

H

Half adders, 56

History of HDLs, 1–3

M

Memory map model, 135, 136

Modern digital design flow, 8

Multiplexer design by hand, 36, 41

Multiplexers, 36, 41

N

Non-volatile memory, 136

O

One-hot binary encoder design by hand, 34, 35

One-hot binary encoder modeling in Verilog, 34, 35

One-hot decoder modeling in Verilog, 30, 31

P

Place and route, 8

R

Random-access memory (RAM), 136

Read cycle, 135, 136

Read-only memory (ROM), 136

Read/write (RW) memory, 136

Ripple carry adders (RCA), 58

S

Semiconductor memory, 135

Sequential access memory, 136

7-segment decoder design by hand, 31

7-segment decoder modeling in Verilog, 33

T

Technology mapping, 8

V

Verification, 6

Verilog

- always blocks, 66
- arrays, 16
- behavioral modeling techniques
 - agents on a bus, 109, 110
 - counters, 129–131
 - D-flip-flops, 103, 104
 - D-flip-flop with enable, 106, 107
 - D-flip-flop with preset, 105, 106
 - D-flip-flop with reset, 104, 105
 - D-latches, 103
 - encoding styles, 118
 - finite state machines, 113
 - next state logic, 115
 - output logic, 116
 - registers, 107, 108
 - shift registers, 108, 109
 - state memory, 115
 - state variables, 114
 - up counters with enables, 131
 - up counters with loads, 131, 132
 - up counter, 129
 - up counter with range checking, 130
- casex statements, 77
- casez statements, 77
- compiler directives, 20
 - include, 20, 21
 - timescale, 20
- continuous assignment, 23, 24
- continuous assignment with conditional operators, 38
- continuous assignment with delay, 43, 44
- continuous assignment with logical operators, 29
- counters, 129, 130
- data types, 13
- disable, 79
- drive strength, 14
- finite state machines, 113
- forever loops, 77
- for loops, 78
- gate level primitives, 53, 54
- history, 2
- if-else statements, 74, 75
- initial blocks, 66

- net data types, 14
 - number formatting
 - binary, 16
 - decimal, 16
 - hex, 16
 - octal, 16
 - operators, 23
 - assignment, 23
 - bitwise logical, 24
 - bitwise replication, 27
 - Boolean logic, 25
 - concatenation, 26
 - conditional, 26
 - numerical, 27
 - precedence, 28
 - reduction, 25
 - relational, 25
 - parameters, 20
 - procedural assignment, 65
 - procedural blocks, 65
 - repeat loops, 78
 - resolution, 14
 - sensitivity lists, 67, 68
 - signal declaration, 19
 - statement groups, 73
 - structural design and hierarchy, 56–61
 - explicitly port mapping, 51
 - gate level primitives, 53, 54
 - instantiation, 51
 - positional port mapping, 52, 53
 - user defined primitives, 54, 55
 - system tasks, 80
 - file I/O, 81, 82
 - simulation control, 83
 - text I/O, 80
 - test benches, 89
 - user defined primitives, 54, 55
 - value set, 14
 - variable data types, 15
 - vectors, 15
 - while loops, 77
- Volatile memory, 136

W

Write cycle, 135, 136

Y

Y-chart, 5