List of Examples and Exercises

Chapter 2: Code Structure
Example 2.1: Compare-add circuit
Example 2.2: D-type flip-flop (DFF)
Example 2.3: Registered comp-add circuit
Example 2.4: Generic address decoder

Exercise 2.1: Multiplexer
Exercise 2.2: Logic gates
Exercise 2.3: Registered logic
Exercise 2.4: Basic VHDL data types
Exercise 2.5: Type STD_LOGIC

Chapter 3: Data Types
Example 3.1: Tri-state buffer
Example 3.2: Circuit with ‘don’t care’ outputs
Example 3.3: Unsigned /signed multiplier #1
Example 3.4: Unsigned /signed multiplier #2
Example 3.5: Slicing a 1Dx1D array of integers
Example 3.6: Slicing a 1Dx1D array of bit vectors
Example 3.7: Slicing a 2D array of bits
Example 3.8: Multiplexer with 1Dx1D PORT
Example 3.9: Recommended signed multiplier implementation
Example 3.10: Legal versus illegal assignments

Exercise 3.1: Possible data types #1
Exercise 3.2: Possible data types #2
Exercise 3.3: Types and operators in the package standard
Exercise 3.4: Types and operators in the package std_logic_1164
Exercise 3.5: Types and operators in the package numeric_std
Exercise 3.6: Types and operators in the package numeric_std std_logic_arith
Exercise 3.7: Operators in the package std_logic_signed
Exercise 3.8: Integer versus enumerated types
Exercise 3.9: Possible packages
Exercise 3.10: Subtypes
Exercise 3.11: Multi-bit tri-state buffer
Exercise 3.12: Single bit versus bit vector
Exercise 3.13: Hardware optimization with ‘don’t care’ values
Exercise 3.14: 1D array examples
Exercise 3.15: 1Dx1D array examples
Exercise 3.16: 2D array examples
Exercise 3.17: 1Dx1Dx1D array examples
Exercise 3.18: Resetting arrays
Exercise 3.19: Type conversion by type casting
Exercise 3.20: Type conversion by specific functions
Exercise 3.21: Overloaded operator and qualified expression
Exercise 3.22: Array slices
Exercise 3.23: Array dimensionality
Exercise 3.24: Legal assignments #1
Exercise 3.25: Legal assignments #2
Exercise 3.26: Illegal assignments #1
Exercise 3.27: Illegal assignments #2
Exercise 3.28: Array dimensionality #2
Exercise 3.29: Legal versus illegal array slices
Exercise 3.30: Illegal assignments #3

Chapter 4: Operators and Attributes
Example 4.1: Using predefined scalar attributes
Example 4.2: DFF with several event-based attributes
Example 4.3: Specifying device pins with the chip_pin attribute
Example 4.4: Construction of a delay line with the keep attribute
Example 4.5: Keeping redundant registers with preserve and noprune attributes

Exercise 4.1: Logical operators and corresponding types
Exercise 4.2: Arithmetic operators and corresponding types
Exercise 4.3: Comparison operators and corresponding types
Exercise 4.4: Logical operators
Exercise 4.5: Arithmetic operators #1
Exercise 4.6: Arithmetic operators #2
Exercise 4.7: Comparison operators
Exercise 4.8: Shift and concatenation operators
Exercise 4.9: Arithmetic operators for signed types
Exercise 4.10: Attributes of an array type
Exercise 4.11: Attributes of an enumerated type #1
Exercise 4.12: Attributes of an enumerated type #2
Exercise 4.13: The enum_encoding attribute
Exercise 4.14: The chip_pin attribute
Exercise 4.15: The keep attribute
Exercise 4.16: preserve versus keep attributes
Exercise 4.17: Capturing digits from a decimal number

Chapter 5: Concurrent Code
Example 5.1: Multiplexer implemented with operators
Example 5.2: Multiplexer implemented with WHEN and SELECT
Example 5.3: ALU
Example 5.4: Generic address decoder with GENERATE
Example 5.5: COMPONENT instantiation with GENERATE
Example 5.6: DFF implemented with concurrent code
Example 5.7: Recommended adder/subtractor implementation
Example 5.8: Short-pulse generator with the keep attribute
Example 5.9: Short-pulse generator with the LCELL primitive

Exercise 5.1: Circuit with “don’t care” outputs
Exercise 5.2: Dual-edge flip-flop
Exercise 5.3: Generic AND and NAND gates
Exercise 5.4: Generic parity generator
Exercise 5.5: Parity generator with automated pin allocation
Exercise 5.6: Generic binary-to-gray converter
Exercise 5.7: Hamming weight with GENERATE
Exercise 5.8: Binary sorter with GENERATE
Exercise 5.9: ALU with WHEN
Exercise 5.10: Arithmetic circuit with INTEGER
Exercise 5.11: Arithmetic circuit with STD_LOGIC
Exercise 5.12 Barrel shifter with INTEGER and BIT_VECTOR
Exercise 5.13: Barrel shifter with STD_LOGIC_VECTOR
Exercise 5.14: Recommended unsigned adder/subtractor implementation
Exercise 5.15: Recommended unsigned multiplier implementation
Exercise 5.16: Recommended signed multiplier implementation
Exercise 5.17: Recommended unsigned divider implementation
Exercise 5.18: Recommended signed divider implementation
Exercise 5.19: Frequency multiplier with the keep attribute
Exercise 5.20: Generic multiplexer
Exercise 5.21: INOUT bus
Exercise 5.22: INOUT versus BUFFER
Exercise 5.23: Floating-point adder
Exercise 5.24: Floating-point adder, subtracter, multiplier, divider

Chapter 6: Sequential Code
Example 6.1: DFFs with reset and clear
Example 6.2: Basic counter
Example 6.3: Shift register
Example 6.4: Carry-ripple adder
Example 6.5: Leading zeros
Example 6.6: Slow 0-to-9 counter with SSD
Example 6.7: Incomplete combinational design

Exercise 6.1: Latch and flip-flop
Exercise 6.2: Gray counter
Exercise 6.3: Registered multiplexer
Exercise 6.4: Generic registered multiplexer
Exercise 6.5: Shift register with load
Exercise 6.6: Hamming weight with LOOP
Exercise 6.7: Binary sorter with LOOP
Exercise 6.8: Signal generator
Exercise 6.9: Switch debouncer
Exercise 6.10: Two-digit timer
Exercise 6.11: Frequency meter (with SSDs)
Exercise 6.12: Programmable signal generator
Exercise 6.13: Programmable signal generator with frequency meter
Exercise 6.14: Digital wall clock (with SSDs)
Exercise 6.15: Data serializer

Chapter 7: SIGNAL and VARIABLE
Example 7.1: Counter with SHARED VARIABLE
Example 7.2: SIGNAL versus VARIABLE usage
Example 7.3: Counters with SIGNAL and VARIABLE
Example 7.4: DFF with q and qbar
Example 7.5: Over-registered counter
Example 7.6: Dual-edge flip-flop
Example 7.7: Generic Hamming weight with concurrent code

Exercise 7.1: SIGNAL versus VARIABLE #1
Exercise 7.2: SIGNAL versus VARIABLE #2
Exercise 7.3: Latches and flip-flops  
Exercise 7.4: Combinational versus sequential circuits #1  
Exercise 7.5: Concurrent versus sequential code circuits #2  
Exercise 7.6: Over-registered counter  
Exercise 7.7: Registered circuits  
Exercise 7.8: Shift register  
Exercise 7.9: Frequency divider with VARIABLE  
Exercise 7.10: Frequency divider with SIGNAL  
Exercise 7.11: Frequency divider with symmetric phase  
Exercise 7.12: Generic AND with concurrent code  
Exercise 7.13: Generic NAND with concurrent code  

Chapter 8: PACKAGE and COMPONENT  
Example 8.1: PACKAGE with FUNCTION and deferred CONSTANT  
Example 8.2: Circular shift register with COMPONENT  
Example 8.3: Parity detector with COMPONENT and GENERIC MAP  
Example 8.4: Shift register with COMPONENT and GENERATE  
Example 8.5: Latch implemented with a guarded BLOCK  

Exercise 8.1: Mux with COMPONENT and GENERATE  
Exercise 8.2: Circular shift with COMPONENT #1  
Exercise 8.3: Circular shift with COMPONENT #2  
Exercise 8.4: Parity detector with COMPONENT and GENERIC MAP  
Exercise 8.5: Adder with COMPONENT and GENERATE  
Exercise 8.6: Synchronous counter with COMPONENT  
Exercise 8.7: Synchronous counter with COMPONENT and GENERATE  
Exercise 8.8: Tapped delay line with COMPONENT and GENERIC MAP  
Exercise 8.9: Programmable signal generator with frequency meter  

Chapter 9: FUNCTION and PROCEDURE  
Example 9.1: FUNCTION max in an ARCHITECTURE  
Example 9.2: FUNCTION order_and_fill in a PACKAGE  
Example 9.3: FUNCTION slv_to_integer in an ENTITY  
Example 9.4: PROCEDURE min_max in a PACKAGE  
Example 9.5: Overloaded “+” operator  
Example 9.6: Non-overloaded “AND” operator  

Exercise 9.1: ASSERT statement #1  
Exercise 9.2: ASSERT statement #2  
Exercise 9.3: Function integer_to_slv  
Exercise 9.4: Function shift_logical_left  
Exercise 9.5: Function my_not  
Exercise 9.6: Function bcd_to_ssd  
Exercise 9.7: Function binary_to_gray  
Exercise 9.8: Procedure mean_and_median  
Exercise 9.9: Procedure equal_length

10. Simulation with VHDL Testbenches  
Example 10.1: Writing values to a file  
Example 10.2: Reading values from a file  
Example 10.3: Stimuli generation  
Example 10.4: Type I testbench for a registered mux  
Example 10.5: Type II testbench for a registered mux  
Example 10.6: Type IV testbench for a registered mux  
Example 10.7: Type IV testbench with a RECORD type  
Example 10.8: Type IV testbench with a data file
***************
Exercise 10.1: Generation of periodic stimuli
Exercise 10.2: Generation of non-periodic stimuli
Exercise 10.3: Writing to a file
Exercise 10.4: Reading from a file
Exercise 10.5: Type I testbench for a LUT-based design
Exercise 10.6: Type II testbench for a LUT-based design
Exercise 10.7: Type III testbench for a LUT-based design
Exercise 10.8: Type IV testbench for a LUT-based design with a record type
Exercise 10.9: Type IV testbench for a LUT-based design with a file
Exercise 10.10: Type I testbench for a binary-to-gray converter
Exercise 10.11: Type I testbench for an address decoder
Exercise 10.12: Type IV testbench for an address decoder
Exercise 10.13: Type I testbench for a carry-ripple adder
Exercise 10.14: Type IV testbench for a carry-ripple adder
Exercise 10.15: Type I testbench for a shift register
Exercise 10.16: Type IV testbench for a shift register
Exercise 10.17: Type I testbench for a Fibonacci series generator
Exercise 10.18: Type II testbench for a Fibonacci series generator
Exercise 10.19: Type IV testbench for a Fibonacci series generator

11. VHDL Design of State Machines
Example 11.1: Vending-machine controller
Example 11.2: Glitch-free vending-machine controller
Example 11.3: Zero-to-nine counter
Example 11.4: Car alarm with bypasses prevented by a flag
Example 11.5: Car alarm with bypasses prevented by additional states
Example 11.6: FSM with embedded timer
Example 11.7: Traffic-light controller
Example 11.8: Pushbutton sequence detector

***************
Exercise 11.1: Gray-encoded counter
Exercise 11.2: Johnson-encoded counter
Exercise 11.3: One-hot-encoded counter
Exercise 11.4: Zero-to-nine counter
Exercise 11.5: Signal generator #1
Exercise 11.6: Signal generator #2
Exercise 11.7: Signal generator #3
Exercise 11.8: ASCII sequence detector
Exercise 11.9: Preventing state-bypass with a flag #1
Exercise 11.10: Preventing state-bypass with additional states #1
Exercise 11.11: Preventing state-bypass with a flag #2
Exercise 11.12: Preventing state-bypass with additional states #2
Exercise 11.13: Timed FSM #1
Exercise 11.14: Timed FSM #2
Exercise 11.15: FSM with embedded timer #2
Exercise 11.16: Traffic-light controller #1
Exercise 11.17: Traffic-light controller #2
Exercise 11.18: Pushbutton sequence detector
Exercise 11.19: Car speed monitor

12. VHDL Designs with Basic Displays
Section 12.2 Basic LED/SSD/LCD Driver
Section 12.3 Playing with a Seven-Segment Display
Section 12.4 Frequency Meter (with LCD)
Section 12.5 Digital Clock (with SSDs)
Section 12.6 Quick-Finger Game (with LEDs and SSDs)

Exercise 12.1: Counter with SSD display
Exercise 12.2: Counter with LCD display
Exercise 12.3 Playing with a Seven-Segment Display
Exercise 12.4: Frequency meter (with SSDs)
Exercise 12.5: Digital clock (with LCD)
Exercise 12.6: Quick-finger game with blinking LEDs
Exercise 12.7: Quick-finger game with linear time reduction
Exercise 12.8: Timer (with LED and SSDs)
Exercise 12.9: Timer (with LED and LCD)
Exercise 12.10: Genius game (with LEDs and SSDs)
Exercise 12.11: Frequency Meter Testing Procedure
Exercise 12.12: Frequency Meter with a Synchronizer
Exercise 12.13: Frequency Meter with Gray Counter

13. VHDL Design of Memory Circuits
Section 13.2 Implementing Bidirectional Buses
Section 13.4a ROM Implemented with Regular VHDL Code
Section 13.4b ROM Implemented with an Initialization File
Section 13.4c ROM Implemented with a Vendor-Specific Function
Section 13.5a RAM Implemented with Regular VHDL Code
Section 13.5b RAM Implemented with a Vendor-Specific Function
Section 13.5c RAM Implemented in a User SRAM Block
Section 13.6 External Memory Interfaces

Exercise 13.1: Bidirectional bus
Exercise 13.2: INOUT versus BUFFER mode
Exercise 13.3: COE versus MIF files
Exercise 13.4: Synchronous ROM
Exercise 13.5: ROM implemented with a HEX file #1
Exercise 13.6: ROM implemented with a HEX file #2
Exercise 13.7: RAM implemented with a HEX file
Exercise 13.8: ROM implemented in an user SRAM block
Exercise 13.9: Synchronous RAM
Exercise 13.10: External memory interface

14. VHDL Design of Serial Communications Circuits
Section 14.2a Design of a Fast Serializer
Section 14.3a Design of a PS2 Keyboard Interface
Section 14.4a Design of an \textit{I}^{2}\textit{C} Interface for an EEPROM Memory
Section 14.5a Design of an SPI Interface for an FRAM Memory
Section 14.6a Design of a TMDS Encoder

Exercise 14.1: Synchronous versus asynchronous communication
Exercise 14.2: Generic serializer
Exercise 14.3: FSM-based serializer
Exercise 14.4: Design of a shift-register-based serializer
Exercise 14.5: Deserializer
Exercise 14.6: Codewords for PS2 keyboard
Exercise 14.7: Analysis of PS2 keyboard interface
Exercise 14.8: PS2 keyboard interface with LCD
Exercise 14.9: Measuring EEPROM latency with \textit{I}^{2}\textit{C} interface
Exercise 14.10: \textit{I}^{2}\textit{C} interface for an RTC
Exercise 14.11: \textit{I}^{2}\textit{C} interface for an ADC
Exercise 14.12: \textit{I}^{2}\textit{C} interface for a temperature sensor
Exercise 14.13: I²C versus SPI
Exercise 14.14: Generic SPI interface for an FRAM
Exercise 14.15: SPI interface for a Flash memory
Exercise 14.16: SPI interface for an ADC
Exercise 14.17: SPI interface for an RTC
Exercise 14.18: TMDS encoder

15. VHDL Design of VGA Video Interfaces
Section 15.9 Hardware-Generated Image
Section 15.10 Image Generation with a File and On-Chip Memory
Section 15.11 Arbitrary Image Generation with a File and Off-Chip Memory
Section 15.12 Image Equalization with Gamma Expansion

-----------------------------------------------
Exercise 15.1: 800x600x75Hz SVGA interface
Exercise 15.2: Image generation with hardware #1 (banner)
Exercise 15.3: Image generation with hardware #2 (sun in the sky)
Exercise 15.4: Image generation with hardware #3 (filling with green)
Exercise 15.5: Image generation with hardware #4 (rotating bar)
Exercise 15.6: Image generation with hardware #5 (digital clock)
Exercise 15.7: Image generation with hardware #6 (arcade game)
Exercise 15.8: Image generation with a file and on-chip memory #1 (banner)
Exercise 15.9: Image generation with a file and on-chip memory #2 (sun in the sky)
Exercise 15.10: Image generation with a file and off-chip memory (arbitrary picture)

16. VHDL Design of DVI Video Interfaces
Section 16.7 Hardware-Generated Image

-----------------------------------------------
Exercise 16.1: TMDS encoder
Exercise 16.2: Image generation with hardware #1 (banner)
Exercise 16.3: Image generation with hardware #2 (sun in the sky)
Exercise 16.4: Image generation with hardware #3 (filling with green)
Exercise 16.5: Image generation with hardware #4 (rotating bar)
Exercise 16.6: Image generation with hardware #5 (wall clock)
Exercise 16.7: Image generation with hardware #6 (arcade game)
Exercise 16.8: Image generation with a file and on-chip memory #1 (banner)
Exercise 16.9: Image generation with a file and on-chip memory #2 (sun in the sky)
Exercise 16.10: Image generation with a file and off-chip memory (arbitrary picture)

17. VHDL Design of FPD-Link Video Interfaces
Section 17.4 Hardware-Generated Image
Section 17.5 Hardware-Generated Image with Characters

-----------------------------------------------
Exercise 17.1: FPD-Link encoder
Exercise 17.2: Image generation with hardware #1 (banner)
Exercise 17.3: Image generation with hardware #2 (sun in the sky)
Exercise 17.4: Image generation with hardware #3 (filling with green)
Exercise 17.5: Image generation with hardware #4 (rotating bar)
Exercise 17.6: Image generation with hardware #5 (digital clock)
Exercise 17.7: Image generation with hardware #6 (arcade game)
Exercise 17.8: Image generation with a file and on-chip memory #1 (banner)
Exercise 17.9: Image generation with a file and on-chip memory #2 (sun in the sky)
Exercise 17.10: Image generation with a file and off-chip memory (arbitrary picture)