Verilog Coding Questions

1. Code a 2:1 multiplexer using both if-else and case.
2. Write code for a D flip-flop with asynchronous reset.
3. Implement an AND gate using assign and always block.
4. Create a 4-bit binary counter with synchronous reset.
5. Write a 4-to-1 multiplexer.
6. Write a 3-bit up counter with enable and reset.
7. Design a priority encoder.
8. Write a simple shift register.
9. Code a 1-bit comparator.
10. Create a half-adder and full-adder.
11. Write a testbench for a 4-bit counter.
12. Implement a Moore FSM for even-odd detector.
13. Write Verilog code for an 8-bit barrel shifter.
14. Design a universal shift register.
15. Implement a simple ALU (AND, OR, ADD, SUB).
16. Design a binary to Gray code converter.
17. Write code to debounce a push-button input.
18. Create a 2-digit BCD counter.
19. Write a synchronous FIFO.
20. Write a dual-edge triggered flip-flop.
21. Write a Mealy machine for sequence detector (e.g., 101).
22. Implement traffic light controller using FSM.
23. Write an FSM to detect three consecutive 1's in a stream.
24. Design FSM for elevator control.
25. Create FSM for vending machine.

26. Create a ROM using case statement.
27. Write a dual-port RAM.
28. Design an asynchronous FIFO.
29. Implement an LRU cache block (simplified).
30. Code for line buffer for image processing.
31. Code for a frequency divider by 2, 4, 10.
32. Design a pulse width detector.
33. Implement one-pulse generator from a button input.
34. Create a watchdog timer.
35. Create a digital clock in Verilog.
36. Implement clock gating logic.
37. Write a code for handshake protocol.
38. Design a parameterized counter.
39. Implement edge detector.
40. Write a Gray code counter.
41. Create a UART transmitter/receiver (basic).
42. Design an SPI master.
43. Implement I2C protocol FSM.
44. Code a CRC generator.
45. Create a simple AXI-lite interface block.
46. Write a testbench for FSM.
47. Design a register file with 8 registers.
48. Implement pipelined multiplier.
49. Create priority-based arbiter.
50. Write code for pulse synchronizer between clock domains.
51. Code a ring counter.

52. Design a Johnson counter.
53. Write code for binary to BCD converter.
54. Write a GCD calculator.
55. Implement basic MAC unit.
56. Design an FSM for password checker.
57. Create a binary search FSM.
58. Implement a pipeline stall controller.
59. Write a glitch-free multiplexer.
60. Write a code to implement debounce logic.
61. Create an AXI transaction state machine.
62. Design an N-bit carry-lookahead adder.
63. Build a parameterized ALU with flags.
64. Create FSM with multiple outputs and internal states.
65. Implement time-multiplexed display driver.
66. Write synthesizable code for radix-4 booth multiplier.
67. Implement basic instruction decoder for RISC-V.
68. Create an FSM to handle serial data framing.
69. Implement a bus arbiter for 4 masters.
70. Design pipeline bypass logic.
71. Implement round-robin arbiter.
72. Create a memory-mapped register interface.
73. Design a delay line using shift registers.
74. Create an FSM with parameterized state encoding.
75. Implement a reconfigurable counter.
76. Write Verilog code for an N-bit adder using generate statement.
77. Implement an N-bit subtractor with overflow detection.

78. Design a multiplier using shift-add method.
79. Write a carry-save adder.
80. Code a leading zero counter.
81. Design a parity generator and checker.
82. Create a priority encoder with mask input.
83. Design a programmable counter (with variable step size).
84. Write code for signed multiplication using Booth's algorithm.
85. Implement a Wallace Tree multiplier (conceptual level).
86. Implement a 3-stage pipelined multiplier.
87. Design a pipeline register with enable and reset.
88. Create a generic pipeline wrapper module.
89. Write a stall control logic for a pipeline.
90. Implement a hazard detection unit.
91. Code for positive edge detector.
92. Code for negative edge detector.
93. Write logic to detect rising and falling edge simultaneously.
94. Create glitch-free edge detection logic.
95. Write a pulse stretcher module.
96. Generate N DFFs using generate and for.
97. Create a parallel adder using a loop.
98. Design a dynamic priority encoder using a loop.
99. Implement a priority decoder using a loop.
100. Design a shift register with variable length using parameterization.
101. Implement a basic SPI Slave module.
102. Write a Verilog UART Baud Rate Generator.
103. Code a basic I2C Master FSM with ACK/NACK logic.

104. Write a simple APB slave interface.

105. Implement a Wishbone-compliant register interface.

106. Create a single-port synchronous RAM using array.

107. Create dual-port synchronous RAM.

108. Implement memory with byte enable support.

109. Write a content-addressable memory (CAM) model.

110. Implement a block RAM with initialization from a file.

111. Write a testbench to verify a barrel shifter.

112. Create stimulus with random data using $random.

113. Write a self-checking testbench.

114. Generate waveform output with $dumpvars.

115. Write a scoreboard module in testbench.

116. Implement fixed-priority arbiter for N requesters.

117. Implement round-robin arbiter using FSM.

118. Design a weighted round-robin scheduler.

119. Code an interrupt controller.

120. Create an FSM-based traffic arbiter for buses.

121. FSM to handle input handshake protocol with ACK/NACK.

122. FSM to parse a serial data stream with headers and footers.

123. FSM with timeouts (using counters).

124. FSM for automatic door controller.

125. FSM to detect pause-resume-reset command sequence.

126. Implement Binary to Gray and Gray to Binary converter.

127. Design a binary to Excess-3 converter.

128. Create a BCD to Binary converter.

129. Implement 7-segment display driver for BCD input.

130. Write a floating-point encoder (IEEE 754 format, basic).
131. Write code for a crossbar switch (4x4).
132. Implement a bitonic sorter (4 elements).
133. Code a simple cache controller FSM.
134. Design a basic DMA controller.
135. Create an AXI-lite Write FSM.
136. Implement a 2-flop synchronizer.
137. Write a multi-bit synchronizer.
138. Design CDC safe pulse synchronizer.
139. Create reset synchronizer logic.
140. Code handshake synchronizer between two clock domains.
141. Design a power-on reset generator.
142. Implement low-power clock gating cell.
143. Write power sequencing logic for subsystems.
144. Code programmable glitch-free mux for clock sources.
145. Create brown-out detector logic in Verilog.
146. Add assertions for a FIFO (full, empty, overflow).
147. Create a timeout check for FSM lock-up.
148. Write an assertion to check illegal state in FSM.
149. Implement a simple event counter for debug.
150. Add a logging module that logs transitions to file.