Verilog Coding Exercises

1. Write a verilog code for 4:1 MUX using a case statement
2. Implement a D Flip-Flop with asynchronous reset
3. Design a 4-bit ripple carry adder using full adders
4. Create an always-on blinking LED using a clock divider
5. Write a Verilog code to detect a 101 pattern in a serial input
6. Implement a 3 -bit counter with an enable signal
7. Write a Verilog code for a 4 -bit Gray code counter
8. Design a priority encoder for 8 inputs
9. Implement an 8-bit shift register with parallel load
10. Write Verilog code for an 8 -bit even parity generator
11. Implement a 3 -bit up/down counter
12. Design a Mealy FSM to detect a 110 pattern in a serial input
13. Design an 8-bit barrel shifter
14. Write a Verilog module to generate a PWM signal
15. Implement a 16-bit Linear Feedback Shift Register (LFSR)
16. Create a Verilog code for a signed multiplier
17. Write a Verilog code for a traffic light controller using FSM
18. Design a priority arbiter for 4 requests
19. Implement a Verilog module for a binary to BCD converter
20. Write a Verilog code for a dual-port RAM.
21. Implement a pipelined 4-stage RISC processor
22. Implement 4x4 Matrix Multiplication
23. Implement a parameterized FIFO with full and empty flags
24. Design an AXI4 -Lite Slave Interface.
25. Implement a DDR3 Memory Controller
26. Design a High-Speed Serializer for a 64-bit Parallel Input
27. Implement an Interrupt Controller with Priority Encoding
28. Design a Configurable ALU with 4 Operations: ADD, SUB, AND, OR.
29. Create a 16-bit barrel shifter that performs logical left, right, and circular shifts based on a control signal
30. Implement a Dual-Port RAM with Independent Read/Write Ports
31. Write a Verilog code for a 4-bit binary up/down counter with enable and reset
32. Create a Traffic Light Controller for a 4 way Intersection.
33. Design a CRC (Cyclic Redundancy Check) Generator for Error Detection
34. Design a Verilog module for a synchronous FIFO queue with parameterized depth.
35. Design a Pipelined Multiplier with 4 Stages

36. Write a Verilog code to implement an LFSR based pseudo-random number generator
37. Create a Generic FIFO with Configurable Depth and Data Width
38. Implement a 32-bit Floating-Point ALU Supporting Addition, Subtraction, and Multiplication
39. Design a High-Speed Serializer for a 64-bit Parallel Input
40. Implement an AXI4 Lite Protocol Slave Interface
41. Design a dual-port RAM with separate read and write ports and asynchronous operations.
42. Create a clock divider that divides the input clock frequency dynamically based on an input division factor
43. Implement a Verilog code for a priority encoder with enable.
44. Create a Verilog module that implements a digital lock using a finite state machine
45. Implement a Verilog module for a 4-bit Gray Code Counter.
46. Design a MUX that cycles through 8 input and outputs each channel sequentially.
47. Implement a Verilog module to compute the Fibonacci sequence iteratively
48. Implement a Verilog module that detects and counts the no. of rising edges in a signal.
49. Design a Verilog module to implement an N-bit multiplier using the array multiplication method.
50. Write a Verilog code to implement a configurable PWM generator with adjustable duty cycle.