Learning Verilog: Essential Practice Questions

Basics of Verilog Syntax and Semantics

1. Write a module that performs bitwise AND, OR, and XOR on two 4-bit inputs.

2. Demonstrate the use of assign statement vs always block with a 2-bit adder.

3. Implement a 4-bit signed and unsigned comparator.

4. Use wire and reg properly in a design with combinational logic.

Procedural Blocks and Behavioral Modeling

5. Design a 4-bit synchronous counter using always block.

6. Model a simple traffic light controller using case statements.

7. Use if-else and case to build a 2-to-4 decoder and 4-to-1 multiplexer.

8. Write a behavioral model of a priority encoder.

Sequential Logic and Flip-Flops

9. Write a D Flip-Flop with synchronous reset.

10. Implement a T Flip-Flop with enable.

11. Create a 3-bit up-down counter with parameterized max value.

12. Design a 4-bit shift register with parallel load and serial output.

Testbenches and Simulation

13. Write a testbench for a 4-bit adder with multiple test cases.

14. Design a testbench that tests a counter with reset and enable.

15. Use $monitor and $display to debug signal values in a testbench.

16. Create self-checking testbenches for a multiplexer module.

Parameterization and Reusability

17. Create a parameterized N-bit adder.

Learning Verilog: Essential Practice Questions

18. Write a generic counter module with parameters for max count and step size.

19. Use `ifdef and `define macros to control simulation features.

20. Design a configurable-width shift register.

FSMs and Control Logic

21. Design a Moore FSM to detect sequence 101.

22. Write a Mealy FSM to generate output based on input transitions.

23. Model a pedestrian crossing signal controller using FSM.

24. Optimize FSM state encoding for minimal transitions.

Practical Digital Design

25. Build a simple ALU with 4 operations and status flags.

26. Create a serial data receiver with start/stop bit detection.

27. Design a debouncer for push-button input.

28. Model a digital lock system using FSM.