100+ Verilog Practice Problems

Level 1: Basic Combinational Circuits

1. Write a Verilog module for a 2-input AND gate.

2. Write a Verilog module for a 2-input OR gate.

3. Write a Verilog module for a 2-input XOR gate.

4. Implement a 2-to-1 multiplexer using conditional operator.

5. Write a Verilog module for a 1-bit full adder.

6. Implement a 1-bit half subtractor.

7. Create a 2-to-4 decoder.

8. Write a 4-to-2 encoder.

9. Design an 8-bit buffer.

10. Write a Verilog module to swap two 4-bit numbers using a temporary variable.

Level 2: Intermediate Combinational Logic

1. Design a 4-bit ripple carry adder.

2. Implement an 8-bit magnitude comparator.

3. Design a 4-bit binary to Gray code converter.

4. Create a priority encoder for 4 inputs.

5. Implement a 3-to-8 decoder using case statements.

6. Write a 4-bit barrel shifter (left rotate).

7. Design a simple ALU that supports AND, OR, ADD, SUB.

8. Implement an 8-bit multiplexer with 4 inputs.

9. Create a module that checks for even parity in 8-bit input.

10. Write a Verilog module for a 4-bit comparator using structural modeling.

Level 3: Basic Sequential Logic

1. Design a positive edge-triggered D flip-flop.

2. Create a T flip-flop using a D flip-flop.

3. Write a 4-bit synchronous binary counter.

4. Write a 4-bit asynchronous counter.

5. Implement a modulo-10 counter.

6. Create a sequence detector for '1011'.

7. Implement a shift register (Serial-In-Parallel-Out).

8. Design a JK flip-flop using behavioral modeling.

9. Create a frequency divider by 2 using flip-flop.

10. Write a module for clocked SR latch.

Level 4: FSM and Control Logic

1. Design a Moore FSM to detect '1101' pattern.

2. Design a Mealy FSM for sequence '1001'.

3. Create a traffic light controller for a two-way road.

4. Design an elevator controller for 3 floors.

5. Implement a vending machine for a product costing Rs.10 (accept Rs.5, Rs.10).

6. Build a pedestrian crossing signal with FSM.

7. Create a simple CPU instruction decoder.

8. Design a finite state machine to detect 3 consecutive 1s.

9. Implement a 3-bit LFSR (Linear Feedback Shift Register).

10. Create a PWM signal generator using FSM.

Level 5: Memory and Register File Design

1. Design a 4x4 RAM with read/write enable.

2. Create a register file with 8 registers of 8 bits each.

3. Design a dual-port RAM.

4. Write a single-port synchronous RAM using behavioral style.

5. Create a read-only memory (ROM) initialized with values.

6. Implement a FIFO buffer with full and empty detection.

7. Build a stack with push and pop logic.

8. Design a memory mapped register interface.

9. Write a Verilog module to simulate a cache line.

10. Create a simple instruction memory for RISC-V.

Level 6: Pipelining and Advanced Sequential Logic

1. Implement a 3-stage pipeline (Fetch, Decode, Execute).

2. Design a pipelined 4-bit ALU.

3. Write a forwarding unit to resolve data hazards.

4. Create a branch predictor module.

5. Design a hazard detection unit.

6. Implement a pipeline register between each stage.

7. Write testbench to verify stall conditions.

8. Create a scoreboard for tracking instruction dependencies.

9. Implement a control unit for a simple pipeline.

10. Write a bypass logic module for ALU inputs.

Level 7: Processor Design Projects

1. Design a 4-instruction mini CPU (ADD, SUB, LOAD, STORE).

2. Implement an accumulator-based processor.

3. Design a simple RISC datapath and control logic.

4. Create an 8-bit MIPS-style ALU and register file.

5. Implement control logic for ALU operations.

6. Build a program counter with jump and branch control.

7. Design a fetch-decode-execute FSM for a simple processor.

8. Create a register file with write-back stage.

9. Implement a small instruction memory for your CPU.

10. Simulate a full cycle of instruction execution in testbench.