#100 days of rtl coding

Certainly! Here's a list of 100 Verilog coding questions, one for each day of your #100DaysOfRTL challenge. These questions cover a wide range of Verilog topics and difficulty levels to help you progressively improve your Verilog coding skills:

**Days 1-10: Verilog Basics**

1. Write a Verilog module for a 2-input AND gate.
2. Implement a 4-input OR gate using Verilog.
3. Create a Verilog module for a 3-to-8 decoder.
4. Design a 4-bit binary adder in Verilog.
5. Write Verilog code for a 3-input multiplexer.
6. Implement a 4-bit ripple-carry adder in Verilog.
7. Create a Verilog module for a D flip-flop.
8. Write a Verilog module for a 4-bit binary counter.
9. Implement a simple 2-state finite state machine (FSM) in Verilog.
10. Design a 4-to-1 multiplexer using both behavioral and dataflow modeling styles in Verilog.

**Days 11-20: Sequential Logic** 11. Write Verilog code for a T flip-flop.

1. Implement a 3-bit synchronous up-counter in Verilog.
2. Create a Verilog module for a 4-bit shift register.
3. Design a 3-state FSM with Moore outputs in Verilog.
4. Write a Verilog module for an edge-triggered D flip-flop with asynchronous reset.
5. Implement a 4-bit Johnson counter in Verilog.
6. Design a 3-state FSM with Mealy outputs in Verilog.
7. Create a Verilog module for a 4-bit synchronous down-counter.
8. Write Verilog code for a ring counter.
9. Implement a 4-bit gray code counter in Verilog.

**Days 21-30: Advanced Combinational Logic** 21. Design a 4-to-16 line decoder using case statements in Verilog.

1. Create a Verilog module for a 7-segment display decoder.
2. Implement a 4-bit BCD to 7-segment decoder in Verilog.
3. Write Verilog code for a 4-bit magnitude comparator.
4. Design a 4-input priority encoder in Verilog.
5. Implement a 4-to-1 multiplexer with an enable input in Verilog.
6. Create a Verilog module for a 16-bit barrel shifter.
7. Write Verilog code for a 2:1 priority multiplexer.
8. Implement a 16-bit parallel-to-serial converter in Verilog.
9. Design a 4-bit parallel adder-subtractor in Verilog.

**Days 31-40: Memories and Storage Elements** 31. Write Verilog code for a simple 1-bit register.

1. Create a Verilog module for a 4-bit shift register with parallel load.
2. Implement a synchronous 4x4 RAM module in Verilog.
3. Design a 4-bit asynchronous FIFO buffer in Verilog.
4. Write Verilog code for a 4-bit synchronous up-down counter.
5. Implement a 16x4 ROM module using behavioral modeling in Verilog.
6. Create a Verilog module for a dual-port RAM.
7. Design a 4-bit asynchronous pulse synchronizer in Verilog.
8. Write Verilog code for a 4x4-bit multiplier.
9. Implement a 4-bit stack with push and pop operations in Verilog.

**Days 41-50: State Machines** 41. Design a Mealy state machine for a simple traffic light controller in Verilog.

1. Implement a sequence detector state machine in Verilog.
2. Create a Verilog module for a 4-state UART receiver state machine.
3. Write Verilog code for a Moore state machine that counts the number of 1's in a binary sequence.
4. Design a vending machine controller using a Mealy state machine in Verilog.
5. Implement a state machine for a basic elevator controller in Verilog.
6. Create a Verilog module for a simple game using a state machine.
7. Write Verilog code for a traffic light controller with pedestrian signals.
8. Design a state machine for a simple microwave oven controller in Verilog.
9. Implement a 3-state finite state machine that toggles between three LED patterns in Verilog.

**Days 51-60: Arithmetic and Data Paths** 51. Create a Verilog module for a 4-bit ripple-carry adder-subtractor.

1. Design a 4-bit ALU (Arithmetic Logic Unit) in Verilog.
2. Write Verilog code for a 4x4-bit multiplier using Booth's algorithm.
3. Implement a barrel shifter with variable shift amount in Verilog.
4. Create a Verilog module for a 4-bit shift-and-add multiplier.
5. Design a 16-bit carry-lookahead adder in Verilog.
6. Write Verilog code for a 4x4-bit Wallace Tree multiplier.
7. Implement a 16-bit pipelined multiplier in Verilog.
8. Create a Verilog module for a 16-bit adder-subtractor with overflow detection.
9. Design a 32-bit floating-point adder in Verilog.

**Days 61-70: Synchronization and Timing** 61. Write Verilog code for a simple 2-stage pipeline.

1. Implement a Verilog module for a 4-phase clock generator.
2. Design a Verilog module for a 4-bit synchronous counter with asynchronous reset.
3. Create a Verilog module for a dual-edge triggered flip-flop.
4. Implement a Verilog module for a 4-bit Gray code to binary converter.
5. Write Verilog code for a synchronizer to capture asynchronous inputs.
6. Design a Verilog module for a pulse-width modulator (PWM) generator.
7. Create a Verilog module for a clock divider circuit.
8. Implement a Verilog module for a synchronized FIFO buffer.
9. Design a Verilog module for a simple debouncer circuit.

**Days 71-80: FSM Optimization** 71. Write Verilog code for a 4-state FSM using a case statement.

1. Implement a Mealy state machine for a traffic light controller with optimized state transitions.
2. Design a Verilog module for a vending machine controller with minimal states.
3. Create a Verilog module for a sequence detector FSM with reduced state count.
4. Implement a Moore state machine for a binary sequence detector with minimized transitions.
5. Write Verilog code for a state machine with encoded states and efficient state transitions.
6. Design a Verilog module for a simple elevator controller with optimized state logic.
7. Create a Verilog module for a game with a compact and efficient state machine.
8. Implement a state machine for a microwave oven controller with reduced power consumption.
9. Design a Verilog module for a traffic light controller with minimal hardware.

**Days 81-90: Advanced Topics** 81. Write Verilog code for a simple Pulse Width Modulation (PWM) controller.

1. Implement a Verilog module for a finite impulse response (FIR) filter.
2. Design a Verilog module for a digital signal processor (DSP) accumulator.
3. Create a Verilog module for a direct digital synthesizer (DDS).
4. Write Verilog code for a simple UART transmitter and receiver.
5. Implement a Verilog module for a shift register