Problem 1: Logic

Consider the circuit below.

1. What function $F$ of the provided inputs $A$, $B$, and $C$ does the circuit compute?

2. Which input, if any, can be tied to a 0 or 1 to produce a two-input XOR gate? How should it be tied?

3. What should inputs $B$ and $C$ be tied to in order to turn this circuit into a NOT gate (i.e. $F = \bar{A}$)?

4. If this circuit were to be fabricated, what is the minimum number of test vectors that would be needed to ensure that there are no stuck-at-0 faults at the outputs of the 5 logic gates? A stuck-at-0 fault is a defect where once the output of a gate goes to 0, it can never transition to a 1 regardless of its inputs.

Problem 2: Logic Simplification

Refer to this truth table containing 4 inputs ($A$, $B$, $C$, $D$) and 1 output ($Q$).
1. Write an unsimplified sum-of-products expression for $Q$ directly from the truth table.

2. Use a Karnaugh map to write the simplified sum-of-products representation of $Q$.

3. Use a Karnaugh map to write the simplified product-of-sums representation of $Q$.

4. Using the product-of-sums representation, draw the circuit that implements this function using only 2-input AND and OR gates with the minimum number of total gates. How many AND and OR gates are there? You may use inputs and their complements (e.g. $A$ and $\overline{A}$) as inputs to your logic gates.

5. Transform this circuit such that it is made up of only inverters and 2-input NAND gates using the minimum number of total gates. How many inverters and NAND gates are there? Once again, you may use inputs and their complements as inputs to your logic gates.

**Problem 3: FSMs**

In this problem, you will design an FSM for determining whether a number is divisible by 3. The number will be fed in one bit per clock cycle, from most significant to least significant. The output of the FSM should indicate whether the provided bits currently correspond to a number that is divisible by 3.

For example, for input 0, the output should be 1. For the input sequence 1, 1, 0, the output should be 0, 1, 1.

1. Fill in the transitions and associated outputs of the state diagram of this circuit. Populate the transition trigger and output with $X$ if the transition cannot occur. As a hint, the only state you need to store is the remainder of the number when divided by 3.
2. What type of state machine is this?

3. Fill in the blanks in the provided code to implement this state machine. You may assume that the output should be updated at the rising clock edge at which the input is processed (in this case, this allows us to make the output value dependent only the current state, which is also updated at the rising clock edge).

```verilog
module div3 (input wire in, input wire clk, output wire out);
    reg [1:0] state = 0;
    localparam S1 = 0;
    localparam S2 = 1;
    localparam S3 = 2;

    assign out = (___[A]___) ? 1'b1 : 1'b0;

    always @ (posedge clk) begin
        case (state)
            S1:
                begin
                    if (in == 0) ___[B]___;
                    else ___[C]___;
                end
            S2:
                begin
                    if (in == 0) ___[D]___;
                    else ___[E]___;
                end
            S3:
                begin
                    if (in == 0) ___[F]___;
                    else ___[G]___;
                end
        endcase
    end
endmodule
```

4. Suppose we wanted to verify the functionality of our FSM by adding the following assertion to the above `div3` module:
assert property ( 
  @(posedge clk)
  ($past(in) === 1'dx) \[ (state == ___[YOUR CODE HERE]___) \]
);

Complete this assertion by expressing state as an expression of $past(state) and $past(in) given that the value of state is essentially the remainder of the current value when divided by 3.

5. Suppose we wanted to modify the state machine to determine whether a number is divisible by 6 (instead of just divisible by 3). What is the minimum number of total states needed to implement this new FSM as a Mealy machine? Do not include the "Start" state.

6. Implement the state machine from Part 5 in Verilog and simulate it with the input below.

Input: 11011001010111010010101010101010101010101010101010101010101110101

(a) What is the output sequence for this input? Provide your answer as a 64-bit bit string, in the same format as the input. Your output should have 64 digits representing the output after each of the 64 digits of the input are passed to the FSM. As a sanity check, the first 7 digits of your output should be 0010011.

An example output sequence corresponding to an input that is never divisible by 6 is provided below:

0000000000000000000000000000000000000000000000000000000000000000

(b) Attach your Verilog module and testbench. Make sure your module contains an assertion similar to that of Part 4.