PROBLEM 1: GENERAL QUESTIONS
Answer the following questions in a concise way. You are encouraged to look for the answer online. These questions will not be graded for the homework and you are not required to provide a report for them or submit anything, but they might be part of the final exam.

1) Draw the State diagram for the following verilog code.

```verilog
module fsm( clk, rst, inp, outp);
    input clk, rst, inp;
    output outp;
    reg [1:0] state;
    reg outp;
    always @(posedge clk, posedge rst)
    begin
        if( rst )
            state <= 2'b00;
        else
            begin
                case( state )
                    2'b00:
                        begin
                            if( inp ) state <= 2'b01;
                            else state <= 2'b10;
                        end
                    2'b01:
                        begin
                            if( inp ) state <= 2'b11;
                            else state <= 2'b10;
                        end
                    2'b10:
                        begin
                            if( inp ) state <= 2'b01;
                            else state <= 2'b11;
                        end
                    2'b11:
                        begin
                            if( inp ) state <= 2'b01;
                            else state <= 2'b10;
                        end
                endcase
            end
    end
    always @(posedge clk, posedge rst)
    begin
        if( rst )
            state <= 2'b00;
        else
            begin
                case( state )
                    2'b00:
                        begin
                            if( inp ) state <= 2'b01;
                            else state <= 2'b10;
                        end
                    2'b01:
                        begin
                            if( inp ) state <= 2'b11;
                            else state <= 2'b10;
                        end
                    2'b10:
                        begin
                            if( inp ) state <= 2'b01;
                            else state <= 2'b11;
                        end
                    2'b11:
                        begin
                            if( inp ) state <= 2'b01;
                            else state <= 2'b10;
                        end
                endcase
            end
    end
```
if( rst )
  outp <= 0;
else if( state == 2'b11 )
  outp <= 1;
else outp <= 0;
end
endmodule

2) Explain briefly what is transition and states in FSM and how would you model them in verilog.

3) Write the verilog code for a JK master-slave flip flop and use it to create a T flipflop.

4) List few commonly used encoding techniques in digital design and their advantages.

5) Briefly describe the sources of clock signal in the real world electronic devices, think what keeps your wrist watch or a wall clock ticking.

6) What is the metric to determine the number of flipflops we need for implementing a given FSM?

7) Does better verilog coding help contain the power requirements and area utilized by the hardware? Explain with an example.

PROBLEM 2: FINITE STATE MACHINE: FROM DIAGRAM TO VERILOG
A FSM has a 1-bit input, a 1-bit output and 5 states. The behavior is described by the state diagram below. Implement this FSM in Verilog.

![State Diagram]
PROBLEM 3: FINITE STATE MACHINE: FROM DESCRIPTION TO VERILOG
Implement in Verilog a finite state machine with a 1-bit input and a 1-bit output. The machine outputs two consecutive ‘1’ every time the input pattern ‘1011’ is recognized. When the output is 1, the input is discarded. In all other cases the output is 0.

PROBLEM 4: TRAFFIC LIGHT CONTROLLER
A traffic light controller is a FSM with a 1-bit input and a 2-bit output. Pressing a button gives an input = 1, which turns the light from red (output = 10) to green (output = 11). The light has to be green for 3 clock cycles, then it turns to orange (output = 01) for 2 clock cycles before becoming red again.

PROBLEM 5: BINARY DIVISIBILITY BY 3
Construct an FSM that checks if a binary number is divisible by 3. Specifically, your FSM should take the bits sequentially (i.e. one by one) starting from the LSB and output REM = 1 if the number is not divisible and REM = 0 if it is divisible. Do not try modulo statement in verilog, as it is not synthesizable. Modulo only works for mod2 (This is synthesizable using shift operations).

Checkoff list (to be completed with a tutor, a TA or the instructor before the end of last available office hour on Wednesday 08/31, which starts at 12:30PM and ends at 3.30PM ).

NOTE: before starting the checkoff, a student MUST submit its zipped project via TED. The submitted material MUST contain the source code. The checkoff can only be attempted once, so make sure that your solution is working fine.

You can have your projects compiled in advance.
1) Submit the zipped project folder containing ALL of your source code on TED
2) Explain the functionality of the FSM of the problems
3) Show the state diagram
4) Review the Verilog code for the module
5) Review and explain the Verilog code for the testbench
6) Run testbench and explain the results to verify the functionality

Design your testbench to make the checkoff procedure as easy as possible. You should be able to explain to your grader what is the FSM functionality and how you designed your testbench according to that.