Homework 1 –
General Information

1) These are questions from previous midterms and quizzes to help you prepare for the midterm
2) Read Chapters 3, 6, 7, 8, 9, 14 from your textbook to review Combinational Logic and Finite State Machines.
3) Verify the answers to Verilog related questions by testing your code on https://www.tutorialspoint.com/compile_verilog_online.php

You must work by yourself on the homework. You cannot discuss or collaborate with either your lab partner or anyone else in the class or take help from solution manuals/websites.
You will be referred to the SJA if the grader suspects that you have copied the solutions from somewhere without demonstrating any understanding of the question or the underlying subject matter.
So, it is important to show your work very clearly to demonstrate that the work you submitted is yours and not copied from somewhere.

Question 1:
Solve Problems 3.10 and 3:14 from your textbook

Question 2 Implement F using only 2 input NAND gates. Draw the schematic marking a, b, c, and F clearly. You don't have to write any Verilog. Your design cannot use anything else other than 2-input NAND gates. You don't have to optimize your design.

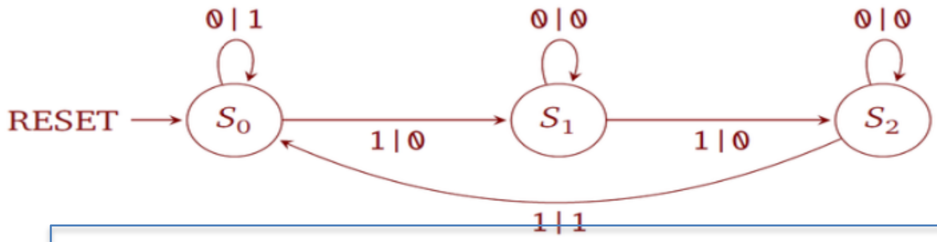
a b c	F
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	0
1 1 0	0
1 1 1	1

Question 3 - Implement the function F described in Question 2 with (a) just one 8:1 mux and (b) with just one 4:1 mux

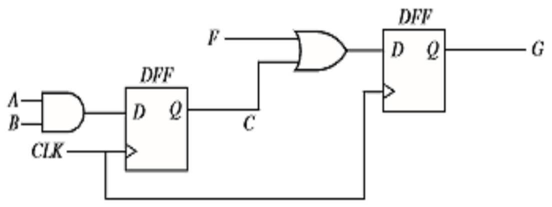
Question 4 Design and implement a synchronous 2-bit **down** counter. The count sequence is 00→11→10→01→00 → .. Use D flip-flops. You must show the transition table, K-maps, and the schematic of the final circuit.

Question 5 Write a high level behavioral description of the 2-bit synchronous counter described above in Verilog The counter should have an asynchronous reset. Add a $monitor task to test the counter.

Question 6
Explain the difference between state assignment and state encoding in a state machine design. Show two difference state encodings and two different state assignments for the state machine shown below.



Question 7 Write a behavioral model for this digital circuits, using an always block. Your model should not contain more than 2 statements inside the always block.



Question 8
Draw the digital circuit for the module whatisit that is described by the Verilog code shown below. Recall "&" denotes logical AND, "|" denotes logical OR operation. Mark all inputs and outputs clearly to receive full credit.

```
module whatisit (y1, y2, clk, rst);
output y1, y2;
input clk, rst;
reg y1, y2;

always @ (posedge clk or posedge rst)
if (rst) y1 <= 0;
else y1 <= y2 & y1;

always @ (posedge clk or posedge rst)
if (rst) y2 <=  0;
else y2 <= y1 | y2;
endmodule
```

Question 9

Draw the circuit represented by the following Verilog process:

```
always @(clk,clr)
begin
    if(clr == 1'b1)
        Q <= 1'b0;
    else if(clk == 1'b0 && CE == 1'b1)
        begin
        if(C == 1'b0)
               Q <= A & B;
        else
               Q <= A | B;
        end
end
```

Question 10- Design a Sequential Multiplier

The combinational multiplier you developed in Lab 1 is cumbersome and impractical if you want to multiply large numbers, say when the number of bits is 32. You can simplify the implementation of A * B by building a sequential circuit that accumulates the shifted values of A based on the value of the least significant bit of B and then shifting B to the right every clock cycle. Design a 32x32 sequential multiplier based on this idea.
Show the schematic of design as clearly as possibly. That should include the high level state diagram of the controller and other components like adders, shifters etc. It should have sufficient detail (such as clk and reset inputs and other control signals to the various blocks) so that one could start writing Verilog code for the design if necessary.

Question 11 - Design a Combinational Circuit
Design a combinational circuit that takes a 3-bit unsigned number X as the input and computes the value of Z, where $Z = 2X^2 + 17X + 8$
You must implement the circuit using exactly one, 4x4 unsigned multiplier you designed in Lab 2.
You are **not allowed** to use any other gates in your design except **one** 4x4 multiplier. Basically you need to figure out what to connect to the inputs of the 4x4 multiplier so that the output of the multiplier is Z.

Question 12 – Writing a Self Checking Testbench to verify that the module prime is correct.

```
module prime (in,  isprime);
input [3:0] in; // 4-bit input
output isprime; // true if input is prime
reg isprime;

always @ (in) begin
case (in)
1, 2, 3, 5, 7, 12, 13 :  isprime = 1'b1;
default: isprime = 1'b0;
endcase
end
endmodule
```