EEC-137
Spring 2019
Dice Game Project

You have been commissioned by a Las Vegas Casino to develop an electronic game.
The game consists of guessing the number of a digital dice.

You may work in groups of 2, but each submission must be done individually and must contain the dice sequence assigned to you.

Game Description:

1) The player will guess a number between 1 and 6 and enter it as a binary number in the "dice_guess" input of the main module.

2) The player will set to 1 the "roll_dice" input and hold it for a random time length. During this time, the electronic dice will roll through the 6 numbers of the dice over and over.

3) The player will then release (set to 0) the "roll_dice" input which will cause the dice to stop rolling. At this point, the "dice_output" should stop on a number between 1 and 6 (just like a real dice).

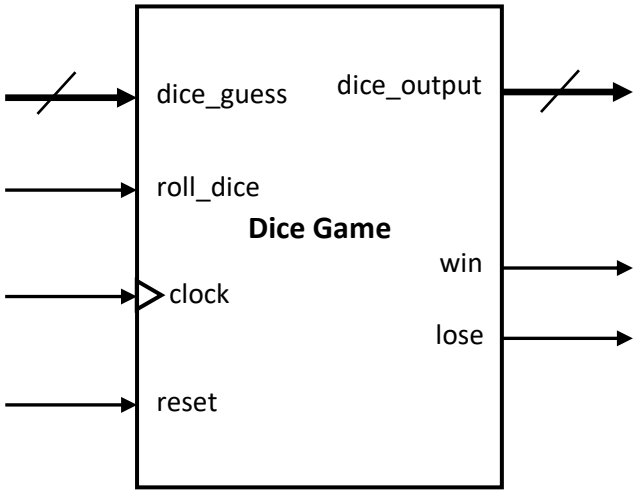
If the number displayed by the dice is the same as the number the player entered, the output will be **win = 1** and **lose = 0**. If the "dice_guess" is different than the "dice_output", then the output will show **win = 0** and **lose = 1**.

4) To start another game, the player must set rest to 1 for at least one clock cycle and then back to 0, then enter a new guess number, and roll the dice again.

Top Module:
1) The "**dice_game**" module should have the following interface which will instantiate the submodule "**dice"** as shown below.

```
module dice_game
(
input    [2:0] dice_guess,
input          roll_dice, clock, reset,
output   [2:0] dice_output,
output         win, lose
);

dice u1(roll_dice, clock, reset, dice_output);
```



Dice module: The dice is basically a counter that goes through the dice sequence. The interface of the "**dice"** module should be as follows

```
module dice
(
input    roll_dice, clock, reset,
output   dice_output
);
```

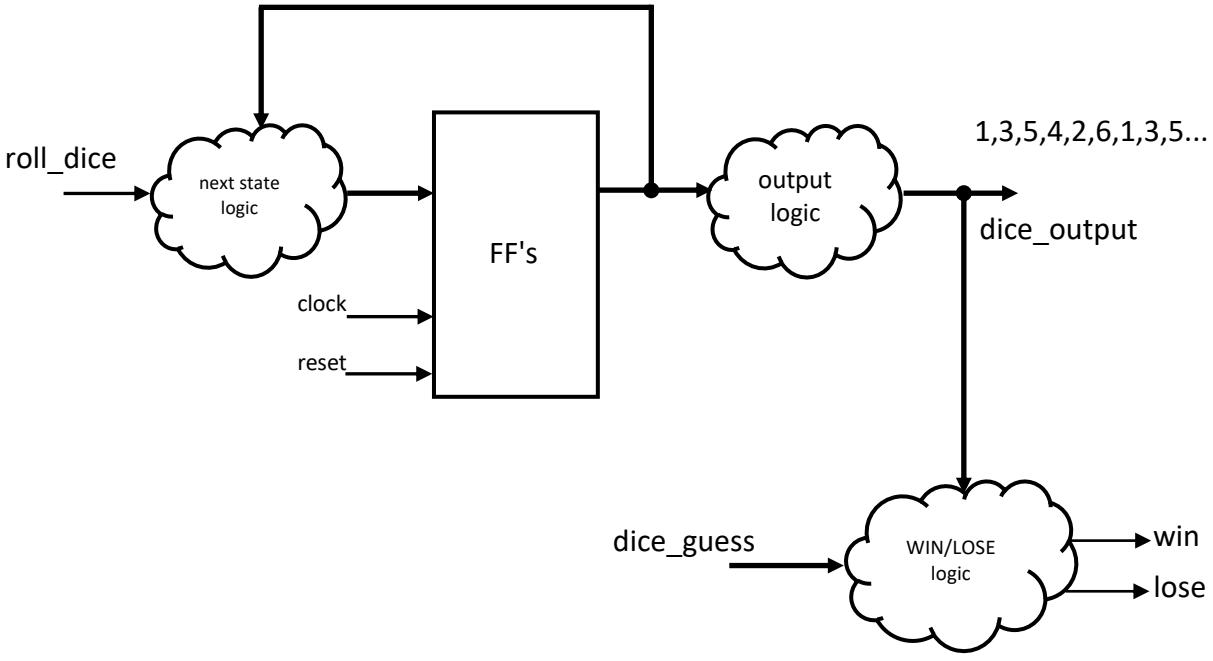
NOTE: The dice roll sequence will be provided to you and it will be unique to each student. The sequence will be available to you in the course website.

For example, one student may get a sequence 6-3-4-5-1-2 and another student may get the sequence 2-5-4-3-1-6, etc.

Win/Lose Logic

The **win/lose** outputs will be generated by simple combinational logic that takes as inputs the **dice_guess** and the **dice_output**. This will produce only 2 outputs: the **win** bit and the **lose** bit.

The following is a block diagram of the overall circuit.



Testbench
Your testbench must ensure that the dice goes through the full dice sequence at least once before it stops.

Your testbench should display the "roll_dice", the "dice_guess", "dice_output", "win", "lose". Please, use the **$time** directive on your testbench to display the time of the events.

Submission:
1) Paper copy (pdf) of FSM design:
    a) Your name and the name of your working partner.
    b) Finite State Diagram
    c) FSM Transition Table
    d) K-maps and equations for the **next state logi**c, and **output logic** and any **win/lose** logic required to compare the dice_guess with the dice_output.
2) Verilog modules and test bench and simv executable.
3) Simulation results (jpegs showing your user name).