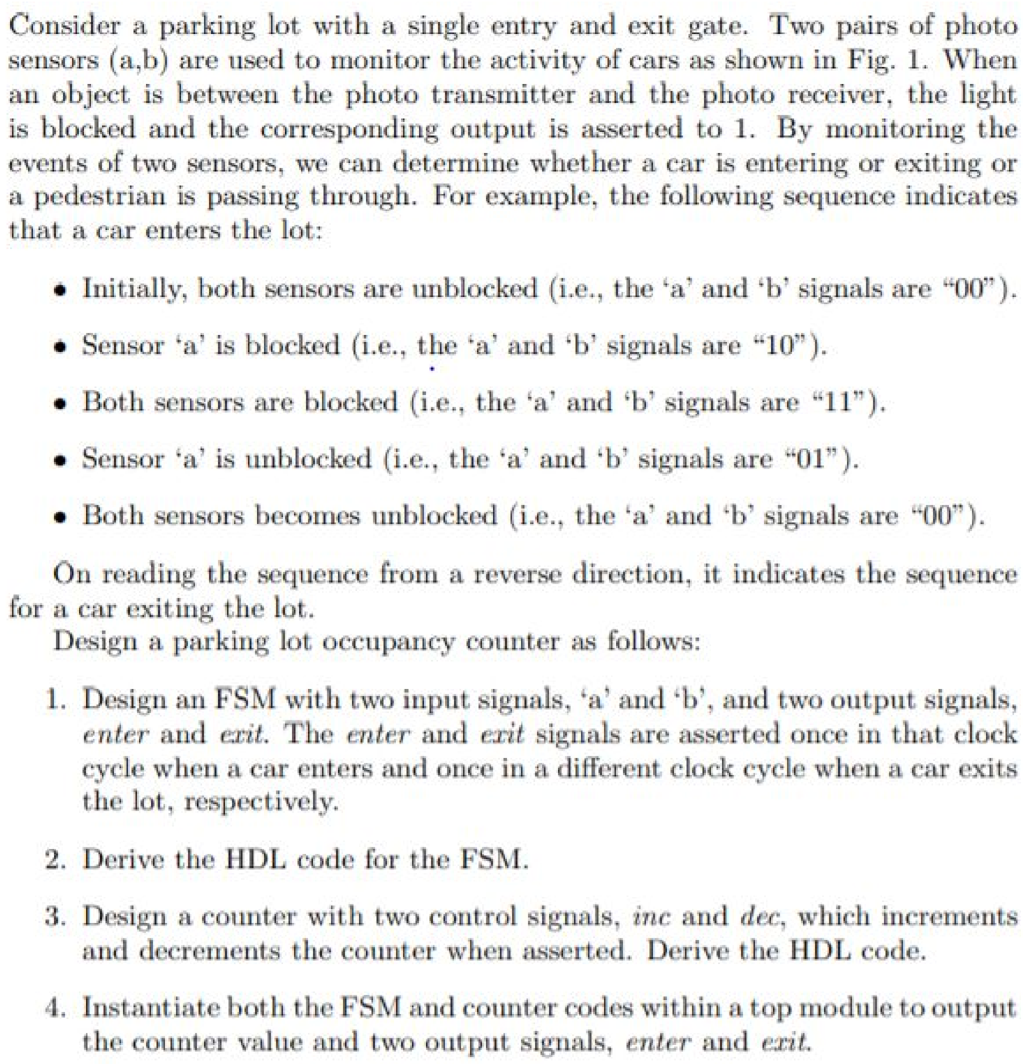
Assignment 4

Instruction: You are required to submit a neatly labeled FSM for the assignment problem with description of each state, and demonstrate the working of the verilog codes and simulation outputs in your respective laptops or computer. Kindly zip your Verilog codes and test bench codes in a folder named by GROUP< NO :> and email it to < nitinkush16@gmail.com > And < sudarshansharma04@gmail.com > before coming to class on 4th February,2020.

1. Problem Statement



Consider a parking lot with a single entry and exit gate. Two pairs of photo sensors (a,b) are used to monitor the activity of cars as shown in Fig. 1. When an object is between the photo transmitter and the photo receiver, the light is blocked and the corresponding output is asserted to 1. By monitoring the events of two sensors, we can determine whether a car is entering or exiting or a pedestrian is passing through. For example, the following sequence indicates that a car enters the lot:

- Initially, both sensors are unblocked (i.e., the 'a' and 'b' signals are "00").
- Sensor 'a' is blocked (i.e., the 'a' and 'b' signals are "10").
- Both sensors are blocked (i.e., the 'a' and 'b' signals are "11").
- Sensor 'a' is unblocked (i.e., the 'a' and 'b' signals are "01").
- Both sensors becomes unblocked (i.e., the 'a' and 'b' signals are "00").

On reading the sequence from a reverse direction, it indicates the sequence for a car exiting the lot.

Design a parking lot occupancy counter as follows:

1. Design an FSM with two input signals, 'a' and 'b', and two output signals, *enter* and *exit*. The *enter* and *exit* signals are asserted once in that clock cycle when a car enters and once in a different clock cycle when a car exits the lot, respectively.
2. Derive the HDL code for the FSM.
3. Design a counter with two control signals, *inc* and *dec*, which increments and decrements the counter when asserted. Derive the HDL code.
4. Instantiate both the FSM and counter codes within a top module to output the counter value and two output signals, *enter* and *exit*.

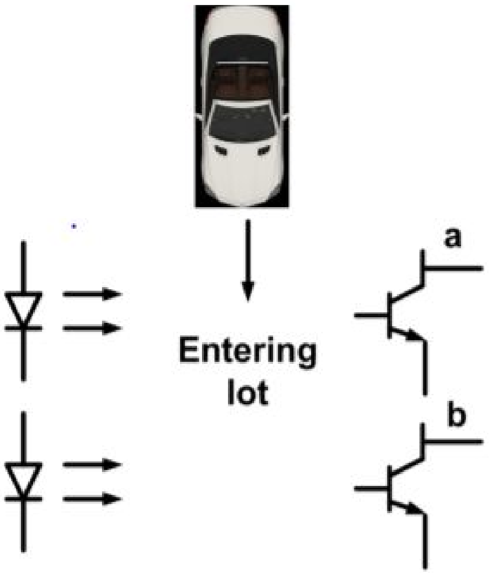


Figure 1: Conceptual diagram of gate sensors

5. Read your inputs 'a' and 'b' from a text file for the purpose of test bench. The text file should have extension .txt and must be saved within the same working directory. You are expected to give around 60 data. You may assume any number of cars or even no car present in the parking lot initially. Also assume that there is space only for a single car to either enter into the lot or make an exit from the lot.

2. Answer the following question

   1. Present a neatly labelled FSM for your circuit by marking every transition clearly.
   2. Have you developed a Moore machine or a Mealy machine for your code?
   3. Comment on how your FSM will undergo structural changes for the other type of machine whose Verilog code you have not written. You need not write the Verilog code for the other type of machine.

   Some Generic Instructions

   1. You may have an initial RESET state.
   2. From the RESET state, 2 sub-FSMs can branch out: one for ENTRY and one for EXIT. The states for a sub-FSM may or may not interact with the states of the other sub-FSMs.
   3. As $\{a, b\}$ are the inputs, from every state S_i, there shall be clear transitions to other states for all possible combinations of $\{a, b\}$.

4. Your input file must have a sequence of states for which some may be invalid. Example: "11" input should not occur just after "00". You have complete freedom to deal with such a state as per your choice.

5. You may or may not treat the following sequence as valid entry: $00 \rightarrow 10 \rightarrow 11 \rightarrow 10 \rightarrow 11 \rightarrow 01 \rightarrow 00$; which indicates that the car makes a partial move back before entering the garage. Design choice is completely yours.

6. The text file from which the sequence of inputs must be read should contain 2 pairs of bits, one in each line to denote the $\{a, b\}$ inputs. It is strongly recommended that you use the text file as your mode of input. On the contrary, you could have used your testbench directly to feed the $\{a, b\}$ inputs. Example for writing the $\{a, b\}$ inputs in the text file:
   00
   10
   11
   and so on.