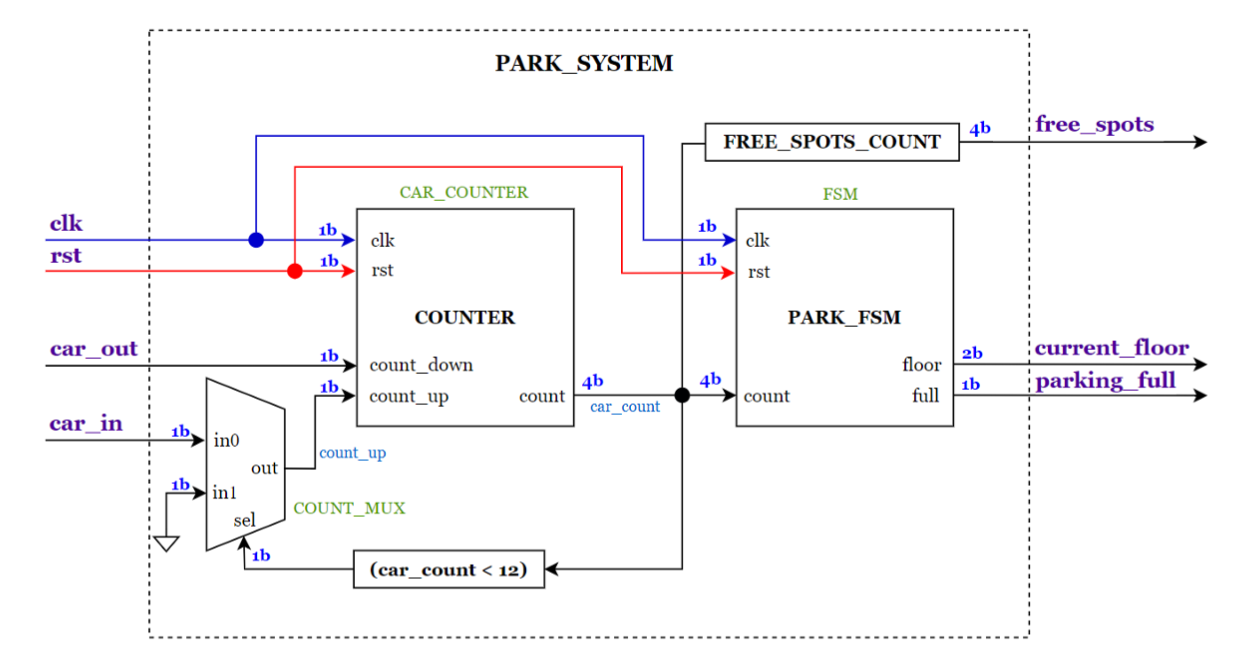
# Simple-Park-System-with-Verilog

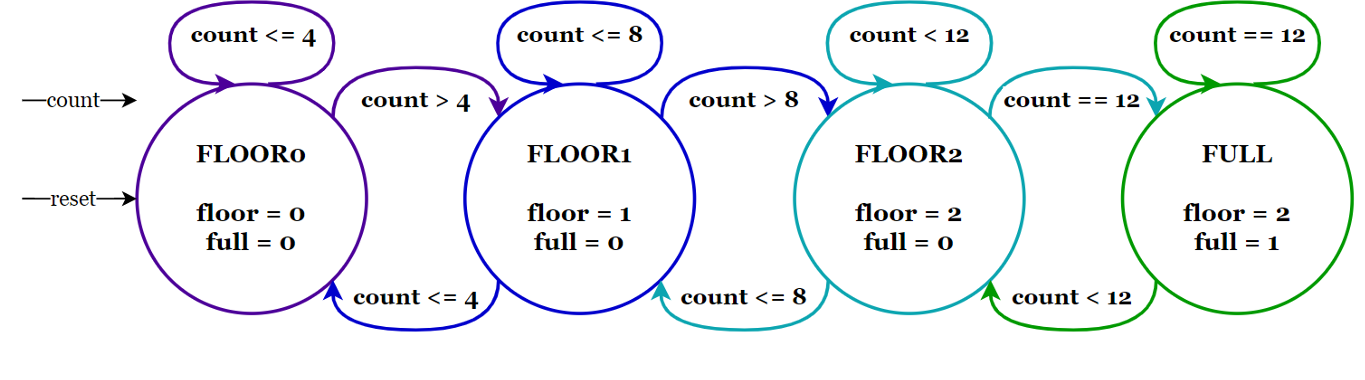
The circuit simulates the operation of a simplified two-storey smart parking system, receiving as input, in addition to clk and rst signals, the control signals car\_in and car\_out.

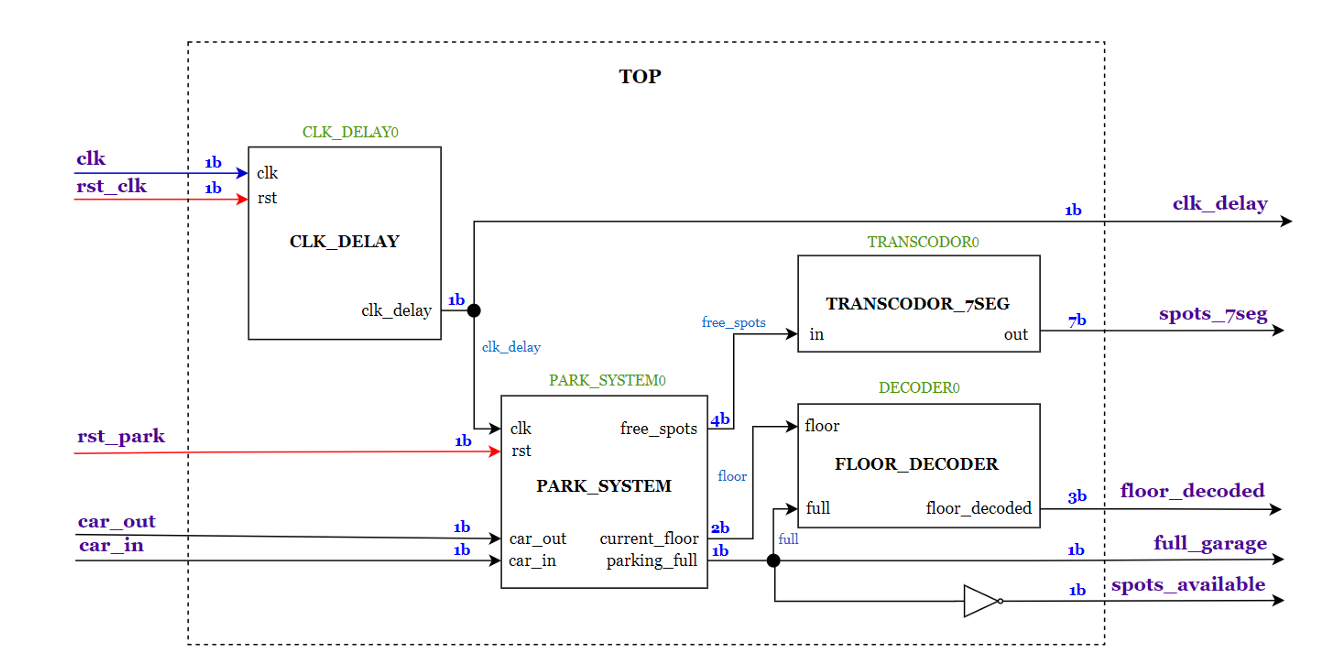
The parking lot has a maximum capacity of 12 spaces, with 4 spaces per floor (ground floor, 1st floor, 2nd floor). The floors are occupied sequentially, and as long as there are free spaces on a particular floor, the next floor remains free. Example: if 6 cars come in a row to be parked, all the parking spaces on the ground floor and two spaces on the 1st floor will be filled. If a car wants to leave, we assume it is the last car parked (last-in-first-out principle), and this means that the ground floor will remain full and one car remains parked on the first floor.

Operating principle: - When car\_in is 1, a car is waiting to be parked, and is to be put in the next available parking space. If the parking lot is full (car\_count == 12, COUNT\_MUX), the car will not be parked and the command is ignored. If there are spaces available (car\_count < 12, COUNT\_MUX), the car is parked and "counted" - When car\_out is 1, the last parked car is removed from the parking lot and the number of occupied parking spaces is decremented and the number of available parking spaces is incremented. For simplicity, the car\_in and car\_out signals are not asserted simultaneously. The output of the circuit provides the status of the full parking lot (parking\_full), the number of available parking spaces (between 0 and 12) and the floor on which the next car will be parked, if there are any available spaces.



PARK\_FSM is an automaton that outputs the occupancy status of the entire parking lot by reaching the maximum number of parked cars (12) and the current floor on which cars can be placed. The graph of the automaton is as follows:





To associate: - the clk bit with the 100MHz clock signal of the FPGA board

- the rst\_clk bit with the BTN0 switch

- the rst\_park bit with the BTN1 switch

- the car\_in bit with the SW0 switch

- the car\_out bit with the SW1 switch

- the clk\_delay bit with the LED- bit with LEDs [0-2]

- bits of floor\_decoded with LEDs [0-2]

- bits of spots\_7sec with the 7 segments in the display

- bit full\_garage with the red LED of RGB0

- bit spots\_available with the green LED of RGB0

To realize these connections, you can create a constraint file or you can link the pins directly in the I/O output menu. For the constraint file, use the syntax:

set\_property -dict { PACKAGE\_PIN PIN\_NAME IOSTANDARD LVCMOS33 } [get\_ports { BIT\_TO\_ASSIGN }]];

Where PIN\_NAME is the pin on the Boolean Board, and BIT\_TO\_ASSIGN is the bit you want to assign to a switch, button, or LED.

