Car Parking System — VHDL Simulation Lab

AUTHOR
Philippe Velha

1 Objective

Design a **Finite State Machine (FSM)** that simulates a simple **car parking access system** using sensors, password inputs, LEDs, and 7-segment displays.
All work will be done **in simulation (no physical hardware)**.

2 Entity Specification

Use exactly this entity definition:

entity Car_Parking_System_VHDL is port(clk : in std_logic; reset_n : in std_logic; – active low, asynchronous front_sensor: in std_logic; back_sensor : in std_logic; password_1 : in std_logic_vector(1 downto 0); password_2 : in std_logic_vector(1 downto 0); GREEN_LED : out std_logic; RED_LED : out std_logic; HEX_1 : out std_logic_vector(6 downto 0); – segments gfedcba HEX_2 : out std_logic_vector(6 downto 0)); end Car_Parking_System_VHDL;

3 FSM States

State	Description	Conditions	Outputs
IDLE	No car present	`front_sensor='0'`	LEDs and displays OFF
WAIT_PASSWORD	Car detected, waiting for password	`front_sensor='1'` and <10 clock cycles elapsed	Red LED ON steady, displays **“En”**
WRONG_PASS	Wrong password entered	After 10 cycles, password != correct	Red LED **blinks**, displays **“EE”**
RIGHT_PASS	Correct password entered	After 10 cycles, password == correct	Green LED **blinks**, displays **“GO”**
STOP	Second car waiting	`front_sensor='1' and back_sensor='1'` while in RIGHT_PASS	Red LED **blinks**, displays **“SP”**

4 Display

Character	gfedcba	Comment
E	`"0000110"`	“E”
n	`"0101011"`	“n”
G	`"0000010"`	use “6” shape for G
O	`"1000000"`	“O”
S	`"0100100"`	“S”
P	`"0001100"`	“P”
OFF	`"1111111"`	all segments off

5 🕹️ Simulation Scenarios

For all the scenarios: Initial state — reset_n=‘0’ → system goes to IDLE.

5.1 Scenario 1: Normal Entry

1. Assert `front_sensor = '1'`
2. Wait 10+ clock cycles
3. Input correct password: `password_1 = "01"`, `password_2 = "10"`
4. Verify: GREEN LED blinks, display shows “GO”
5. Assert `back_sensor = '1'`
6. Verify: System returns to IDLE

5.2 Scenario 2: Wrong Password

1. Assert `front_sensor = '1'`
2. Wait 10+ clock cycles
3. Input wrong password: e.g., `password_1 = "00"`, `password_2 = "00"`
4. Verify: RED LED blinks, display shows “EE”
5. Input correct password
6. Verify: System transitions to RIGHT_PASS

5.3 Scenario 3: Multiple Cars

1. Car 1 enters with correct password → RIGHT_PASS
2. Before `back_sensor = '1'`, assert `front_sensor = '1'` again
3. Verify: System enters STOP state
4. Input correct password for car 2
5. Verify: Both cars processed correctly

5.4 Scenario 4: Reset

1. Put system in any state
2. Assert `reset_n = '0'`
3. Verify: System immediately returns to IDLE
4. All outputs reset

6 ⚠️ Clarifications

Simulation only: no need for debounce or prescale.

Blinking happens at clock frequency (may appear fast in waveform).

Password inputs must be stable during the 10-cycle waiting period.

Use an asynchronous, active-low reset.

The counter width can be small (4 bits is enough for 10 counts).

7 ✅ Deliverables

- Synthesizable VHDL file: Car_Parking_System_VHDL.vhd
- Testbenches that covers the different scenarios
- Simulation waveform showing all states and transitions
- Short report: explain assumptions, FSM design, state diagram and your code (implementation and testbench)