System Description

For

[Garage System]

Submitted to: [Eng. Aya Hegazi]

Submitted by:

[عبدالرحمن كرم محمود, عبدالرحمن محمد طلعت, عبدالرحمن خالد عثمان]

Section 9

Date: [30/4/2023]

Contents

1. Abstract

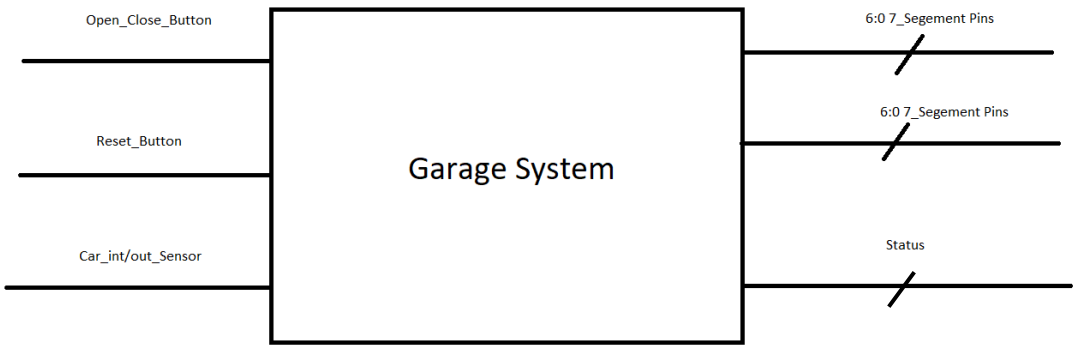
The garage system is designed to manage the entry and exit of cars in a garage.

It has a button to open and close the garage, a counter that increments when a car enters the garage and decrements when a car leaves, and a maximum capacity of 50 cars.

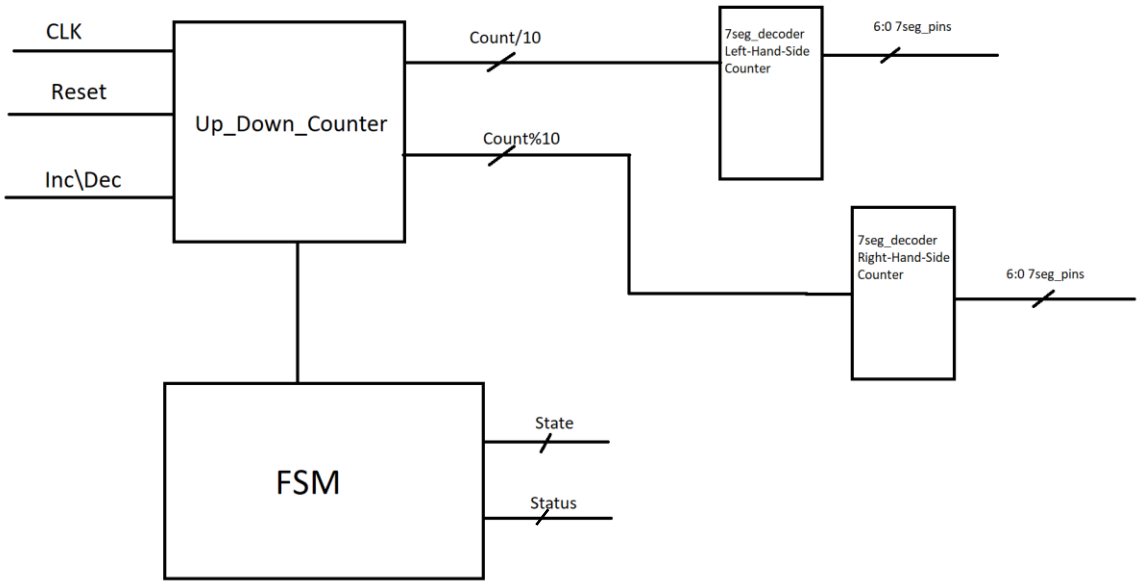
If the garage is full, it will display "GARAGE IS FULL" and will not open until a car leaves.

2. Overview

a. System Icon



b. System Block Diagram

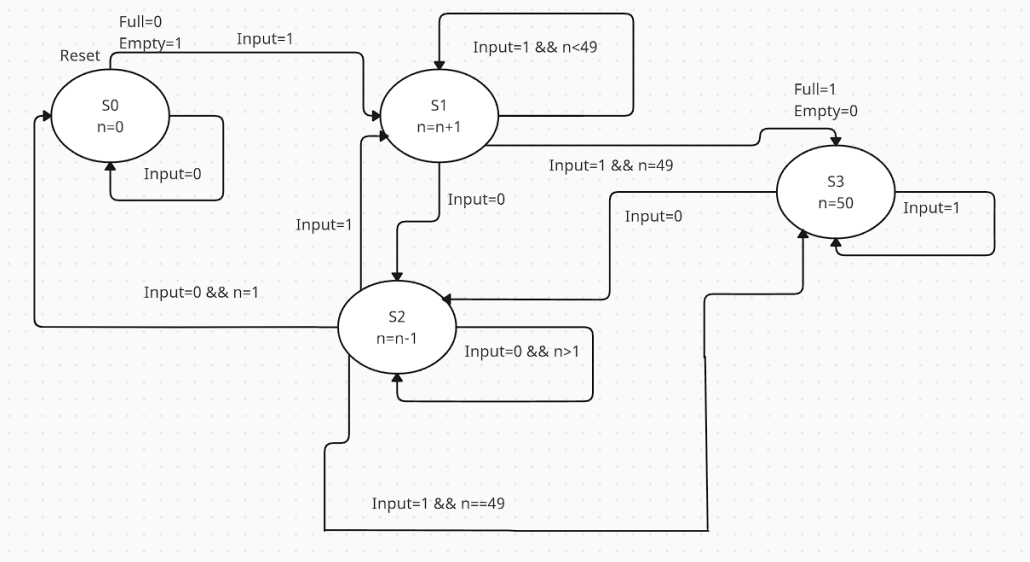


c. Inputs and Outputs Table

Input	Description
Open_close_Button	Button signal indicating a request to park or retrieve a car
Reset_Button	Reset signal to initialize the system
Car_in/out_sensor	Signal indicating the presence of a car in the garage
Output	**Description**
L-H-S 7seg_pins	Left digit of number of cars inside
R-H-S 7seg_pins	Right digit of number of cars inside
Status	Letter indicating the status of the parking system

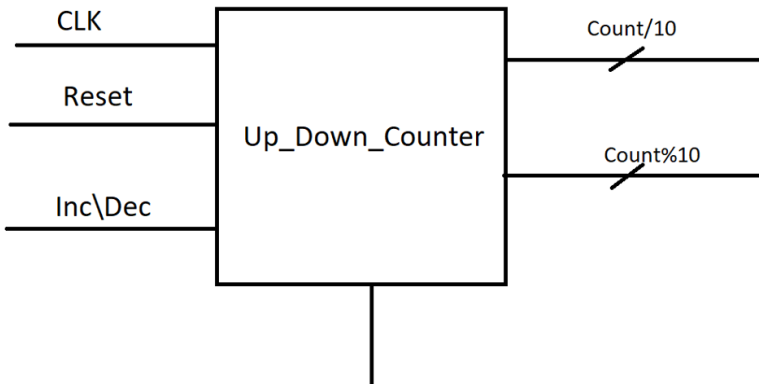
3. Structure

a. FSM Diagram

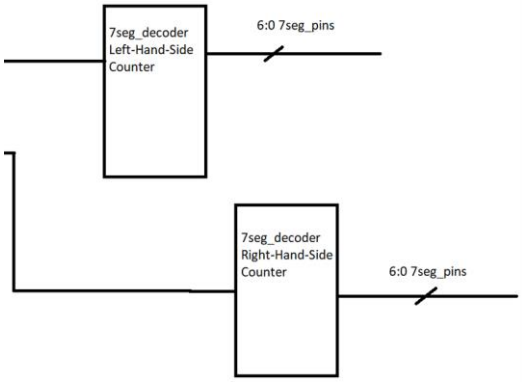


b. Description of Each Module

- Counter: This module takes its input as two numbers
  1: It means that there is a car enter the garage and the counter increment by 1.
  0: It means that there is a car go out of the garage and the counter decrement by 1.
  This counter is up to 50.



- 7-seg decoder: Decodes a 4-bit number into a 7 signals that appears on the 7-segement.



- FSM: Indicates which state is on now.
    1) State 0: This state in which the count of cars is 0 and the garage is open and There are two possibilities: the input may be 1 or 0
       If 1, That means There is a car enters the garage and the counter will increment by 1 and become in state S1.
       if 0, That means There isn't any change because there is no car in the garage and become in the same state (S0).

    2) State 1: This state comes after state 0 or state 1 if a car enters,
       If another car enters it repeats itself, but if the number of cars is 49 it goes to state 3,
       If a car leaves, it goes to state 2.

    3) State 2: This state comes after state 2 after a car leaves, if another car leaves, this state repeats itself.
       If another car enters, it goes back to state 1, but if the number of cars is 49 then it goes to state 3.

    4) State 3: This state comes after state one or state two. It’s when the garage is complete with 50 and if it has input 1 it will be in the same state or input 0 it will go to state 2.

- Top-level system: Combines all the previous modules into one module.
  The modules are:
  module FSM,
  module status_multiplexer,
  module counter,
  module BCD decoder

- Status multiplexer: Switches pins on the third 7-seg to let you know if the garage is full or not.

4. Results

a. Test Strategy

FSM code testbench

```
module FSM_tb();
    wire btn;reg reset,in;wire[5:0] count;wire st;

FSM s(btn,reset,in,count,st);

initial begin
    reset=1;in=1;#100;//0
    reset=0;in=1;#100;//1
    reset=0;in=1;#100;//2
    reset=0;in=0;#100;//1
    reset=1;in=1;#100;//0
    reset=0;in=1;#100;//1

end

endmodule
```

7-segment decoder code testbench

```
module seg_tb();
 reg [5:0] temp; reg [3:0] num;reg [3:0] num2;
 wire[6:0] led;//15wire[6:0] led2;

 initial begin
  temp=15;num=temp/10;num2=(temp%10); #100; // num=1 num2=5

  temp=20;num=temp/10;num2=(temp%10); #100; // num=2 num2=0

  temp=10;num=temp/10;num2=(temp%10); #100; // num=1 num2=0

  temp=6;num=temp/10;num2=(temp%10); #100;  // num=0 num2=6
 end
 seven_segment_decoder test(num, led);
 seven_segment_decoder test2(num2, led2);

endmodule
```

Third 7-segent code to show whether it’s full or empty

```
module multiplexer_tb();

 wire in;
 wire [6:0] out;
 status_decoder g(in,out);

endmodule
```

Top level code testbench

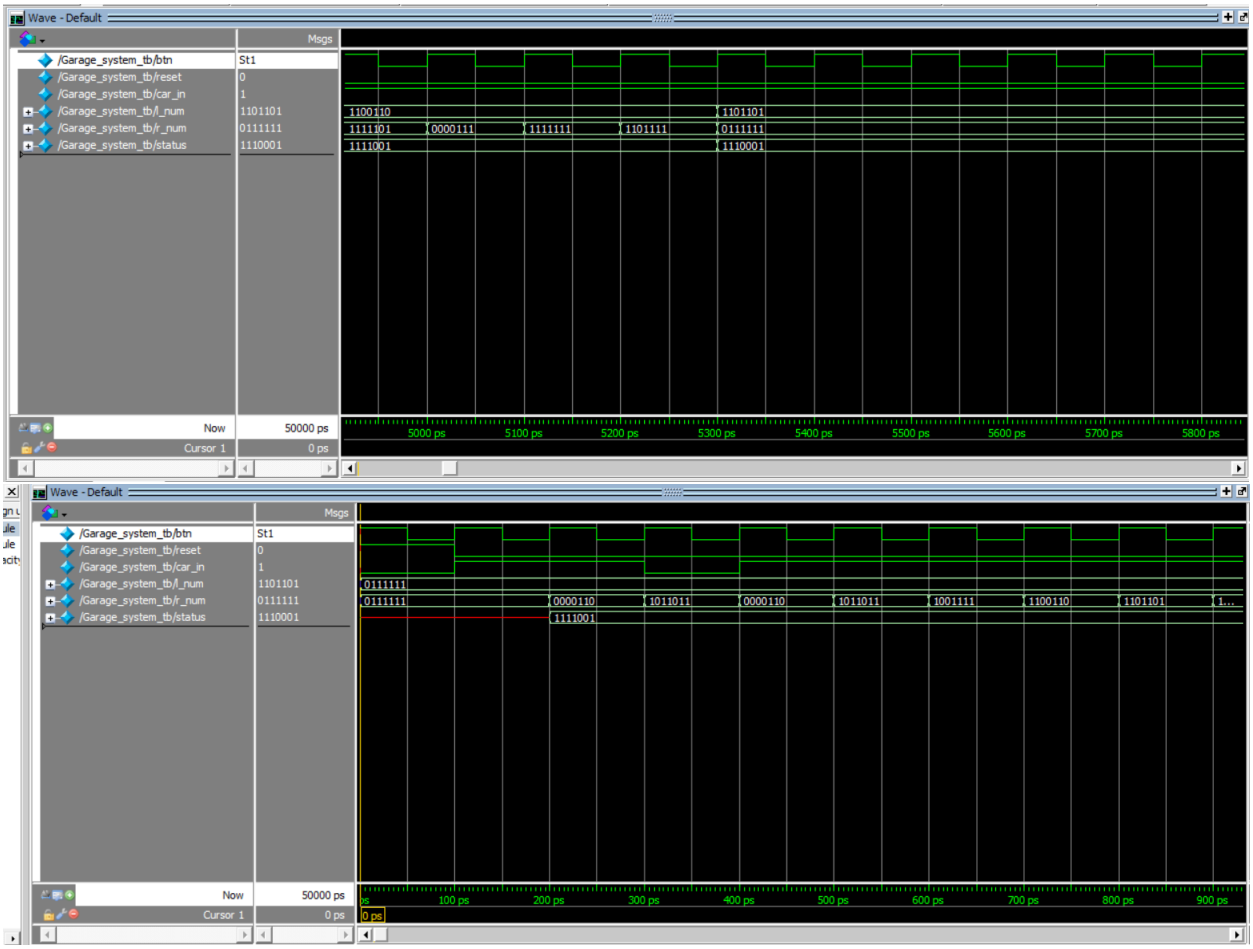
```
module Garage_system_tb();
  wire btn;reg reset, car_in;
  wire [6:0] l_num, r_num;wire[6:0] status;

  Garage_system gs(btn, reset, car_in, status, l_num, r_num);

  initial begin                 //L             R            status
    reset = 1;car_in = 0;#100; //0111111      0111111      1111001
    //////////////////////////
    reset = 0;car_in = 1;#100; //0111111      0000110      1111001
    //////////////////////////
    reset = 0;car_in = 1;#100; //0111111      1011011      1111001
    //////////////////////////
    reset = 0;car_in = 0;#100;
    //////////////////////////
    reset = 0;car_in = 1;#100;
    //////////////////////////
    //reset = 1;car_in = ;#100;
    //////////////////////////
  end

endmodule
```

b. Simulation Results



GitHub link