Counting Cars in Parking Lot

ECE 2700 Final Project – Fall 2020

School of Engineering and Computer Science, Oakland University

Group : Ivan Dawood ivandawood@oakland.edu

Sara Elia saraelia@oakland.edu

Tuan Nguyen Tuannguyen@oakland.edu

Instructors: Daniel Llamocca

December 05, 2020

Abstract:

The purpose of this project is to create a VHDL code and implement it to the Atrix-A7 board to be able to detect the amount of vehicles entering and exiting the parking garage by adding 1, or subtracting 1 and displaying it to the seven segment display. This is important because it would make it easier for people to know if they should enter the garage and try to find parking.

Our project could reduce that if the driver is provided with vacancy information of a parking garage by detecting and counting the number of cars in the garage.

Introduction:

- The purpose of this project is to focus on parking garages and making them faster, less crowded and easier to navigate through.
- During busy times like the holidays, people like to go out and have fun, and usually in busy places like downtown Detroit, there are not many parking areas except on the street and parking buildings.
- The parking spaces on the street are usually timed, which causes people to go into the parking buildings, and that is where people spend the most time navigating through the crowded building just to find out that it is full.
- With the sensors and the seven-segment display added, people would know if the garage is full, and move on to the next one to save time.
- To improve a parking garage, placing a sensor at the entrance and exit will speed up the process because it will allow the sensors to detect the vehicle before reaching a complete stop, and open the gate. It will also do the same when a vehicle is exiting.
- The sensors will also send a signal to the seven-segment display to show the customers if there is space available. If no space is available, the gates will remain shut.

Designment:

1. Electrical Subsystem Overview

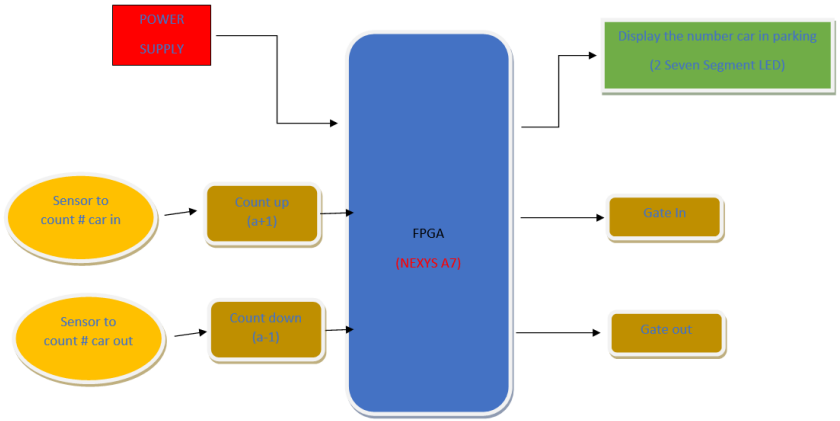


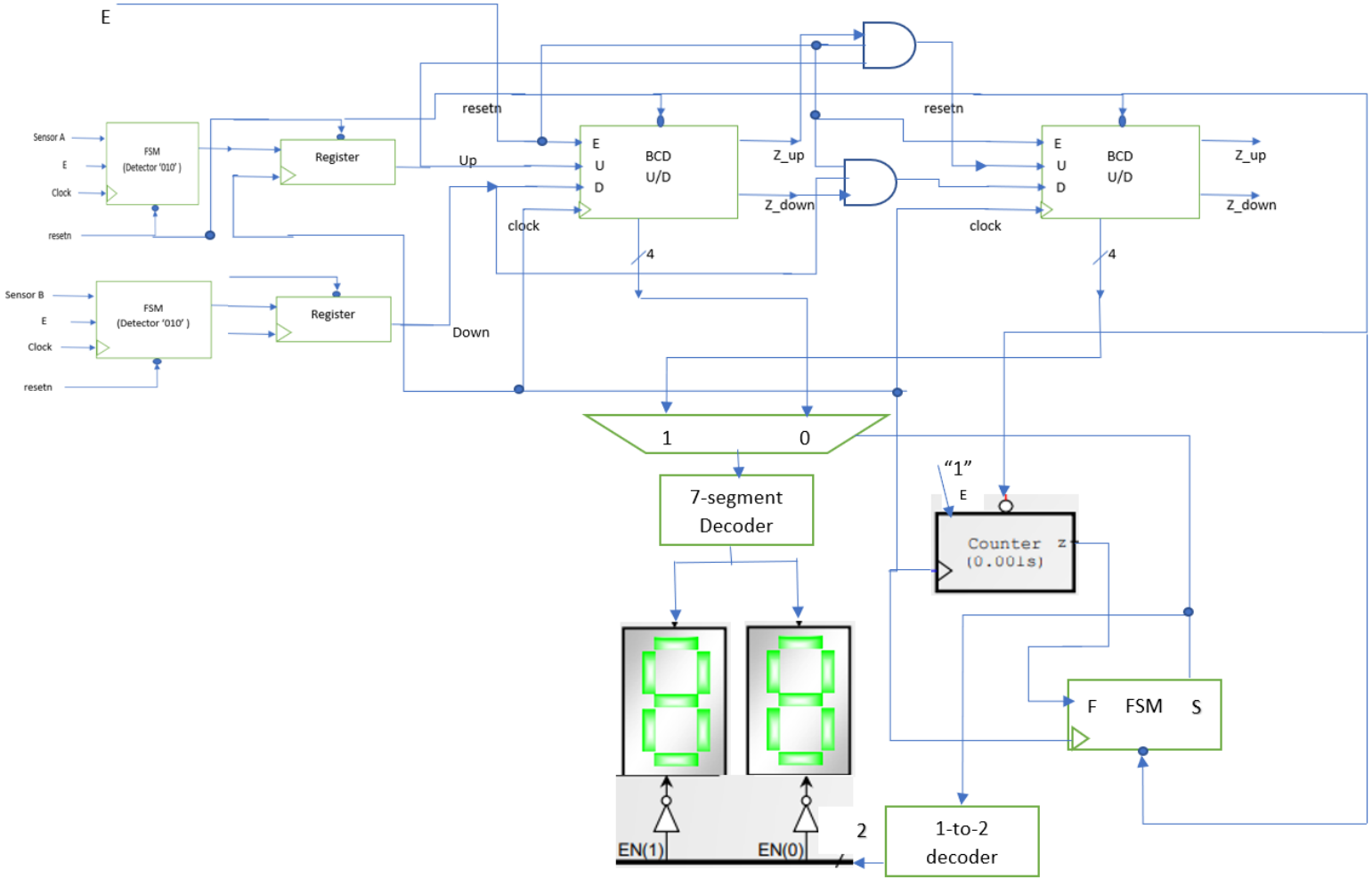
Figure #1: Block Diagram of The Circuit

We have two sensor such as sensor A to detect cars go in and sensor B to detect cars go out the parking.

When sensor A is activated high , the BCD counter will count up (+1). When sensor B is activated high, the BCD counter will count down (-1).

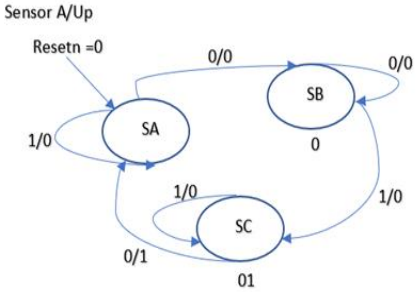
Two sensors will be connected to the Atrix-A7 board to control gate in and gate out.

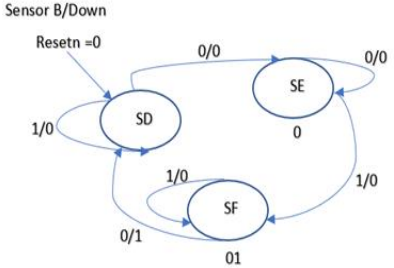
The number of cars in parking will be displayed on two seven segment LED. The maximum which we design for this parking is 99.

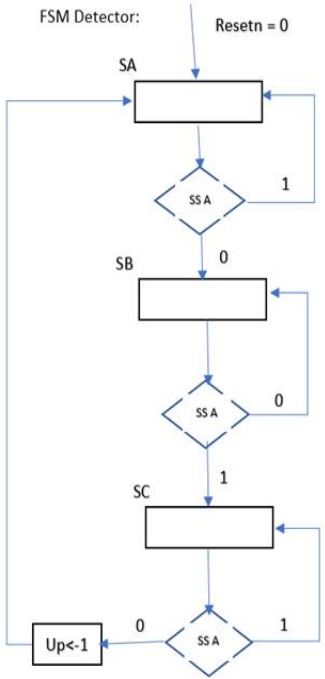

E
Sensor A
E
Clock
resetn
FSM
(Detector '010')
Register
Up
resetn
E
U
D
BCD
U/D
Z_up
Z_down
clock
4
resetn
E
U
D
BCD
U/D
Z_up
Z_down
clock
4
Sensor B
E
Clock
resetn
FSM
(Detector '010')
Register
Down
1
0
7-segment
Decoder
"1"
E
Counter z
(0.001s)
F
FSM
S
EN(1)
EN(0)
2
1-to-2
decoder

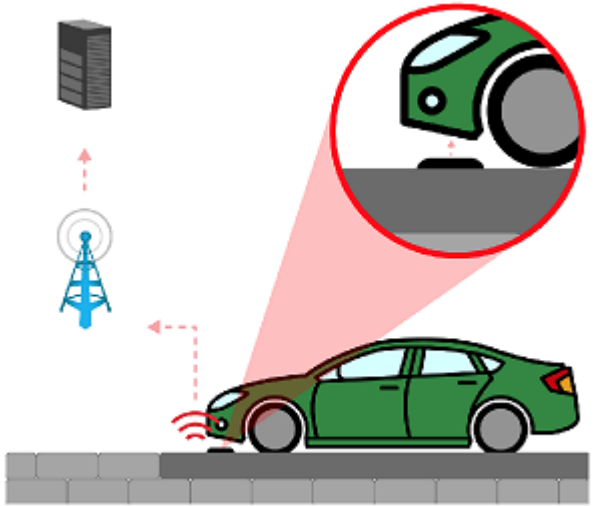
2. Computer Subsystem Overview

A. FMS Detector:

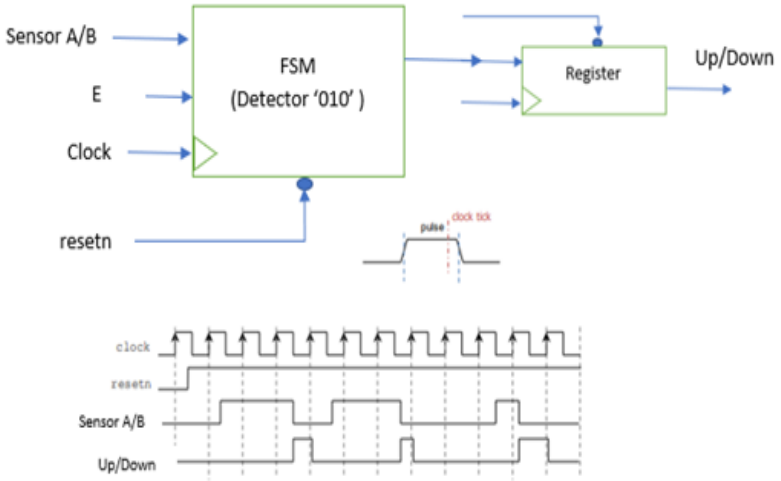






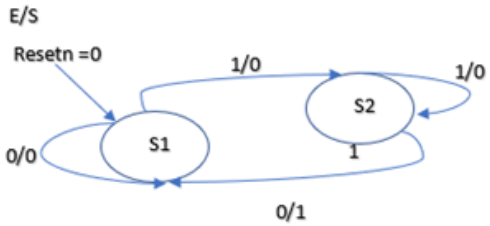


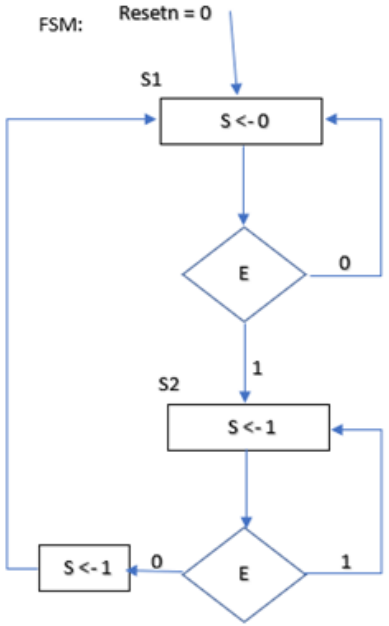
A. FMS Detector(cont'd)

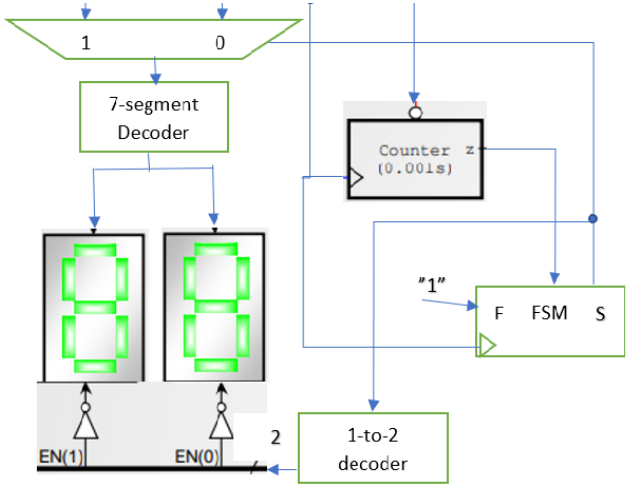


When sensors do not detect any cars entering or exiting the parking garage, they will signal active low '0', and will signal Active high when cars enter or exit. However, when cars rarely move in or out , the sensor will go back '0'. The FSM detector "010" will be used in VHDL code to detect cars in or out, and it will output Up and output Down of the FSMs to connect BCD U/D Counter.

B. FSM:


E/S
Resetn =0
1/0
1/0
S2
S1
0/0
1
0/1


FSM:
Resetn = 0
S1
S <- 0
E
0
1
S2
S <- 1
E
S <- 1
0
1


1
0
7-segment
Decoder
Counter z
(0.001s)
"1"
F FSM S
EN(1)
EN(0)
2
1-to-2
decoder

2. Hardware Overview

Our system is designed to be able to count to a maximum number of 99 vehicles. When BCD count reaches 99, we only allow cars to exit. For those reasons, the gate in will be closed although sensor A is activated, and only the gate out can be opened when sensor B is activated.

When the number of cars is 99, the led middle will light and we know that the parking lots are full. We cannot allow cars to be able to get in.

The Program Code:

Top File:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.math_real.log2;
use ieee.math_real.ceil;

-- This file works for the Nexys-4 Board with eight 7-segment displays
entity Count_Car is
    port (resetn, Up,Down,E, clock: in std_logic; -- resetn: active-low input, Start: active-high input
          segs: out std_logic_vector (6 downto 0);
            en: out std_logic_vector (7 downto 0);
            Full, Gate_in, Gate_out: out std_logic); -- eight 7-segment displays
end Count_Car;

architecture Behavioral of Count_Car is
        component mybcd_udcount is
    port ( clock, resetn, E, Up, Down: in std_logic;
              Q: out std_logic_vector (3 downto 0);
                 z_up ,z_down:out std_logic);
        end component;

    component Counter
    generic (COUNT: INTEGER:= (10**8)/2); -- (10**8)/2 cycles of T = 10 ns --> 0.5 s
    port (E, clock, resetn: in std_logic;
            Q: out std_logic_vector ( integer(ceil(log2(real(COUNT)))) - 1 downto 0);
        z: out std_logic);
    end component;

    component sevenseg
        port (bcd: in std_logic_vector (3 downto 0);
                sevseg: out std_logic_vector (6 downto 0);
                EN: out std_logic_vector(3 downto 0)
                );
    end component;
    signal E_fsm,Up_00, Down_00, Up_0,Up_1,Down_0,Down_1, z_up_0, z_up_1,z_down_0, z_down_1: std_logic;
    signal omux, Q_0, Q_1, ENt: std_logic_vector (3 downto 0);
    signal s: std_logic;
    type state is (S1, S2);
    signal y: state;
    type state_1 is (SA,SB,SC);
        signal y_1: state_1;
    type state_2 is (SD,SE,SF);
        signal y_2: state_2;
```

Test Bench:

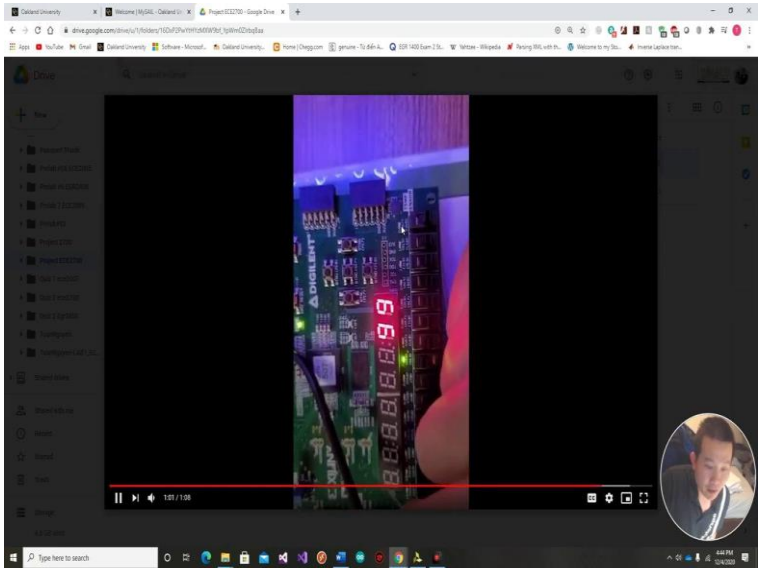
```
 -- Instantiate the Unit Under Test (UUT)
uut: Count_Car PORT MAP (
        resetn => resetn,
        clock => clock,
        E => E,
        Up => Up,
         Down => Down,
        segs => segs,
        EN => EN,
        Gate_in => Gate_in,
        Gate_out => Gate_out,
        Full => Full
      );

-- Clock process definitions
clock_process :process
begin
     clock <= '0';
     wait for clock_period/2;
     clock <= '1';
     wait for clock_period/2;
end process;


-- Stimulus process
stim_proc: process
begin
   -- hold reset state for 100 ns.
   wait for 100 ns;
   resetn <= '0'; E <='0';Up <= '0'; Down <='0';
   wait for clock_period*2; resetn <= '1';  E <='1';
li: for i in 0 to 100 loop
     wait for clock_period*2; Up <= '0';Down <='0';
      wait for clock_period*2; Up <= '1';Down <='0';
end loop;

lj: for j in 0 to 99 loop
       wait for clock_period*2; Up <= '0';Down <='0';
        wait for clock_period*2; Up <= '0';Down <='1';
end loop;
  Up <= '0';Down <='0';
    wait;
  end process;

END;
```

The Simulation:

The Simulation (cont'd):

The Demo Video for The System:

https://drive.google.com/file/d/128w0fjnYghkWQCH9NuSwlKgSbvs2uqvC/view?usp=sharing



Conclusion:

Today, energy and money is being wasted when people are trying to find a parking spot inside garages.

Our project could reduce that if the driver is provided with vacancy information of a parking garage by detecting and counting the number of cars in the garage.

Work cited:

Digital Logic Design VHDL Coding for FPGAs Unit 6- Daniel Llamocca.

Digital Logic Design VHDL Coding for FPGAs Unit 7- Daniel Llamocca.