**LAB NOTE**

**Subject: Digital Design Principles**

**Topic: Parking Indicator**

**Student: Minh Quan Tran**

**May 24th, 2024**

Table of Contents

[1. Objectives 4](#_Toc167480919)

[2. Design 6](#_Toc167480920)

[2.1 Requirement 6](#_Toc167480921)

[2.2 Solution 6](#_Toc167480922)

[3. VHDL and Verilog 6](#_Toc167480923)

[3.1 VHDL code 6](#_Toc167480924)

[3.2 Verilog code 7](#_Toc167480925)

[4. Pin Planner 8](#_Toc167480926)

[4.1 Input and Output 8](#_Toc167480927)

[5. Result 10](#_Toc167480928)

**TABLE OF FIGURES**

[Figure 4‑1: SCEMA5F31C6N board 8](#_Toc167480929)

[Figure 4‑2: SW0 and SW1 Pin No 9](#_Toc167480930)

[Figure 4‑3: LEDR0's Pin No 9](#_Toc167480931)

[Figure 4‑4: HEX's Pin No 9](#_Toc167480932)

[Figure 5‑1: When all 6 levels are available. 10](#_Toc167480933)

[Figure 5‑2: When 1 level is full 10](#_Toc167480934)

[Figure 5‑3: When all levels are full 11](#_Toc167480935)

# Objectives

* Design a parking indicator using Quartus software.
* Write a VHDL and Verilog code from the design.
* Push the code and run to the SCEMA5F31C6N board.

# Design

## Requirement

**Procedure:**  
Consider the following parking control system:  
A parking lot is constructed of 6 levels; each level has 16 parking spaces. At the entrance there is red light indicator. When all the parking spaces (in the 6 levels) are occupied, the red-light indictor is ON and the green Light indicator is OFF, when there is an available parking space at any level the red light indictor is OFF and the green light indicator is ON.  
**Hint:** This is a two stages design, you need to design only the second stage.  
1. Create a new project targeting your FPGA (follow naming criteria)  
2. Create a top level VHDL file containing 6 switch inputs, and 1 led Red  
output.  
3. For the Library section you will need:  
library IEEE;  
use IEEE.STD\_LOGIC\_1164.ALL;  
4. For the Entity section, use the following information:  
SW [0:5] – Input Switches.  
LEDR [2] – Red Output indicator.  
LEDG [3] – Green Output indicator.  
5. In the architecture section, in a process block sensitive to the switch  
inputs, describe combinational logic for the system.  
6. From the Processing menu, run Start->Start Analysis & Synthesis  
7. Assign the LED and SW pins from Assignments -> Pin Planner  
8. Place and route the design using Processing ->Start Compilation.  
9. Download your design to your FPGA.  
10. When your design is working on the FPGA, demonstrate it to the  
instructor.

## Solution

From the Hint, it indicates that there are 2 stages to design,

* First would be the design for each level indicating if there are free slots left.
* Second, will be the design for overall system in which if all levels are occupied then the red LED will light up and the green LED will turn off.

Because it is only required to design for the second stage, so the number of input and output for this system will be:

* Input: 6 levels each will be described as SW0 to SW6 by the board.
* Output: 2 LED, LEDR0 is for green LED and LEDR1 is for red LED.

It states that: “When **all** the parking spaces (in the 6 levels) are occupied, the red-light indictor is ON and the green Light indicator is OFF”, so AND gate with 6 inputs will be best suited for this job.

Also “When there is an available parking space at any level the red-light indictor is OFF, and the green light indicator is ON.” So, the output for red LED is the opposite of green LED.

# VHDL and Verilog

## VHDL code

**library** ieee;

**use** ieee.std\_logic\_1164.**all**;

-- Declaring input and output

**entity** MTran\_Lab3\_VHDL\_ParkIndicator **is**

**port**

(

SW : **in** std\_logic\_vector (5 **downto** 0);

LED: **out** std\_logic\_vector (0 **to** 1);

HEX: **out** std\_logic\_vector (6 **downto** 0)

);

**end** MTran\_Lab3\_VHDL\_ParkIndicator;

-- Describing the relationship between output and input

**architecture** behavioral **of** MTran\_Lab3\_VHDL\_ParkIndicator **is**

**shared** **variable** free\_slot: integer := 0;

**begin**

-- Counting free slot in parking

**process**(SW)

**begin**

-- Resetting free\_slot

free\_slot := 0;

-- Counting free slot in parking

**for** i **in** SW'range **loop**

**if** SW(i) = '0' **then**

free\_slot := free\_slot + 1;

**end** **if**;

**end** **loop**;

-- Displayed number of free slot to 7 segment LED

**case** free\_slot **is**

**when** 0 => HEX <= "1000000";

**when** 1 => HEX <= "1111001";

**when** 2 => HEX <= "0100100";

**when** 3 => HEX <= "0110000";

**when** 4 => HEX <= "0011001";

**when** 5 => HEX <= "0010010";

**when** 6 => HEX <= "0000010";

**when** **others** => HEX <= "1111111";

**end** **case**;

-- Displaying LED

**case**(SW) **is**

**when** "111111" => LED <= "10";

**when** **others** => LED <= "01";

**end** **case**;

**end** **process**;

**end** behavioral;

## Verilog code

**module MTran\_Lab3\_Verilog\_ParkIndicator**(

**input** wire [5:0] SW,

output reg [1:0] LED,

output reg [6:0] HEX

)**;**

// declaring a vector to store 7 Segment LED code

**reg** [6:0] LED\_code[15:0]**;**

integer i = 0;

integer free\_slot = 0;

// initializing declaration

initial **begin**

LED\_code[0] = 7'h40; //0 // can't use 0x40 have to be 7'h or 7'b,...

LED\_code[1] = 7'h79; //1

LED\_code[2] = 7'h24; //2

LED\_code[3] = 7'h30; //3

LED\_code[4] = 7'h19; //4

LED\_code[5] = 7'h12; //5

LED\_code[6] = 7'h02; //6

LED\_code[7] = 7'h78; //7

LED\_code[8] = 7'h00; //8

LED\_code[9] = 7'h10; //9

LED\_code[10] = 7'h08; //A

LED\_code[11] = 7'h03; //b

LED\_code[12] = 7'h46; //C

LED\_code[13] = 7'h21; //d

LED\_code[14] = 7'h06; //E

LED\_code[15] = 7'h0E; //F

**end**

// always block (execute again when there are change in variable)

**always** @(\*)

**begin**

//Resetting variable free\_Slot

free\_slot = 0;

// Check how many free slot remain

for (i = 0; i < 6; i = i + 1)

**begin**

**if** (SW[i] == 0)

**begin**

free\_slot = free\_slot + 1;

**end**

**end**

//Display remaining free slot on 7 segment led

HEX = LED\_code[free\_slot];

//Displaying LED

**case** **(**SW**)**

6'b111111**:** LED = 2'b10**;**

**default:** LED = 2'b01**;**

**endcase**

**end**

endmodule

# Pin Planner

## Input and Output

A close-up of a circuit board

Description automatically generated

Figure 4‑1: SCEMA5F31C6N board

Assigning:

SW[0]: SW0

SW[1]: SW1

SW[2]: SW2

SW[3]: SW3

SW[4]: SW4

SW[5]: SW5

LED[0]: LEDR0

LED[1]: LEDR1

HEX[0]: HEX0[0]

HEX[1]: HEX0[1]

HEX[2]: HEX0[2]

HEX[3]: HEX0[3]

HEX[4]: HEX0[4]

HEX[5]: HEX0[5]

HEX[6]: HEX0[6]

From the DE1\_SoC\_User\_Manual,

A screenshot of a slide switch

Description automatically generated

Figure 4‑2: SW0 and SW1 Pin No

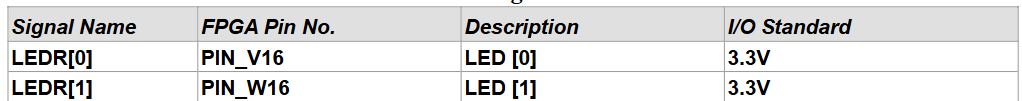


Figure 4‑3: LEDR0's Pin No

 A group of black text

Description automatically generated

Figure 4‑4: HEX's Pin No

# Result

A close up of a circuit board

Description automatically generated

Figure 5‑1: When all 6 levels are available.

All SW are off, mean that all levels are available for parking, 7 segments LED displays ‘6’ mean that there are 6 levels available for parking.

A close up of a circuit board

Description automatically generated

Figure 5‑2: When 1 level is full

When turn on SW3, LEDR1 remained ON indicating there still slot available in this parking slot and the 7 segments LED display ‘5’ which mean there are ‘5’ levels available.

A close up of a circuit board

Description automatically generated

Figure 5‑3: When all levels are full

When all levels are full LEDR1 turns off and LEDR0 turn on indicating it full and the 7 segments LED display ‘0’ which mean no more slots are available.

**REFERENCES**