**LAB NOTE**

**Subject: Digital Design Principles**

**Topic: Seven Segment Display**

**Student: Minh Quan Tran**

**May 31th, 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 9](#_Toc167480925)

[4. Pin Planner 14](#_Toc167480926)

[4.1 Input and Output 14](#_Toc167480927)

[5. Result 16](#_Toc167480928)

**TABLE OF FIGURES**

[Figure 4‑1: SCEMA5F31C6N board 14](#_Toc167480929)

[Figure 4‑2: SW0 and SW1 Pin No 15](#_Toc167480930)

[Figure 4‑3: LEDR0's Pin No 15](#_Toc167480931)

[Figure 4‑4: HEX's Pin No 15](#_Toc167480932)

[Figure 5‑1: When all 6 levels are available. 16](#_Toc167480933)

[Figure 5‑2: When 1 level is full 16](#_Toc167480934)

[Figure 5‑3: When all levels are full 17](#_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

* Use VHDL to implement a binary to 7 Segment decoder.
* Introduction to the VHDL Case statement.
* **Above and beyond:** Create a calculator that can perform basic addition up to 2 digit.

## Solution

To design this system, first need to identify input and output:

* Input will be from SW0 to SW3.
* Output will be the 7 LED from Seven Segment Led.

Then identify what the output will be from the 4 Switches, because the LED is active high so the code for 7 segment code will be:

|  |  |  |
| --- | --- | --- |
| **No** | **Input** | **Output** |
| 0 | 0000 | 100 0000 |
| 1 | 0001 | 111 1001 |
| 2 | 0010 | 010 0100 |
| 3 | 0011 | 011 0000 |
| 4 | 0100 | 001 1001 |
| 5 | 0101 | 001 0010 |
| 6 | 0110 | 000 0010 |
| 7 | 0111 | 111 1000 |
| 8 | 1000 | 000 0000 |
| 9 | 1001 | 001 0000 |
| 10 | 1010 | 000 1000 |
| 11 | 1011 | 000 0011 |
| 12 | 1100 | 100 0110 |
| 13 | 1101 | 010 0001 |
| 14 | 1110 | 000 0110 |
| 15 | 1111 | 000 1110 |

# VHDL and Verilog

## VHDL code

**library** ieee;

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

**use** ieee.numeric\_std.**all**;

-- Declaring input and output

**entity** MTran\_Lab4\_VHDL\_7SegmentLed **is**

**port**

(

SW : **in** std\_logic\_vector (9 **downto** 0);

BT : **in** std\_logic\_vector (3 **downto** 0);

LED: **out** std\_logic\_vector (7 **downto** 0);

HEX0: **out** std\_logic\_vector (6 **downto** 0);

HEX1: **out** std\_logic\_vector (6 **downto** 0);

HEX2: **out** std\_logic\_vector (6 **downto** 0);

HEX3: **out** std\_logic\_vector (6 **downto** 0)

);

**end** MTran\_Lab4\_VHDL\_7SegmentLed;

-- Describing the relationship between output and input

**architecture** behavioral **of** MTran\_Lab4\_VHDL\_7SegmentLed **is**

**shared** **variable** num1\_flag, num2\_flag, add\_flag, err\_num1, err\_num2: Boolean := false;

**shared** **variable** hundred, tenth, unit: integer := 0;

**shared** **variable** err, digit0, digit1, digit2, digit3, num1, num2, result: integer := 0;

-- Function declaration

-- Assignment

**function** SevenSegmentDisplay(Switch : std\_logic\_vector (3 **downto** 0)) **return** std\_logic\_vector **is**

**begin**

**case** (Switch) **is**

**when** "0000" => **return** "1000000";

**when** "0001" => **return** "1111001";

**when** "0010" => **return** "0100100";

**when** "0011" => **return** "0110000";

**when** "0100" => **return** "0011001";

**when** "0101" => **return** "0010010";

**when** "0110" => **return** "0000010";

**when** "0111" => **return** "1111000";

**when** "1000" => **return** "0000000";

**when** "1001" => **return** "0010000";

**when** "1010" => **return** "0001000";

**when** "1011" => **return** "0000011";

**when** "1100" => **return** "1000110";

**when** "1101" => **return** "0100001";

**when** "1110" => **return** "0000110";

**when** "1111" => **return** "0001110";

**when** **others** => **return** "1111111";

**end** **case**;

**end** SevenSegmentDisplay;

-- Displaying Digit

**function** DigitDisplay(Switch : std\_logic\_vector (3 **downto** 0);dig : integer) **return** std\_logic\_vector **is**

**begin**

**if** (dig <= 9) **then** **return** SevenSegmentDisplay(Switch);

**else**

**return** SevenSegmentDisplay("1111");

**end** **if**;

**end** DigitDisplay;

-- Check Digit

**function** DigitCheck(dig0 : integer; dig1 : integer) **return** Boolean **is**

**begin**

**if** (dig0 > 9 **or** dig1 > 9) **then**

**return** true;

**else** **return** false;

**end** **if**;

**end** DigitCheck;

-- End of Function Declaration

-- Main code

**begin**

-- Check SW

Control\_Number: **process**(SW,BT(3))

**begin**

-- Reset

err := 0;

**if** (BT(3) = '0') **then**

HEX3 <= "1111111";

HEX2 <= "1111111";

HEX1 <= "1111111";

HEX0 <= "1111111";

err\_num1 := false;

err\_num2 := false;

**end** **if**;

-- SW9 to change different assignment

**if** (SW(9) = '0') **then**

HEX3 <= "1111111";

HEX2 <= "1111111";

HEX1 <= "1111111";

HEX0 <= SevenSegmentDisplay(SW(3 **downto** 0));

**else**

-- Displaying number 1 and number 2

**if** (add\_flag = false) **then**

**if** (num1\_flag = false) **then**

digit0 := to\_integer(unsigned(SW(3 **downto** 0)));

HEX0 <= DigitDisplay(SW(3 **downto** 0),digit0);

digit1 := to\_integer(unsigned(SW(7 **downto** 4)));

HEX1 <= DigitDisplay(SW(7 **downto** 4),digit1);

-- Check num1

err\_num1 := DigitCheck(digit0, digit1);

-- Turn off HEX2 and

HEX2 <= "1111111";

HEX3 <= "1111111";

**elsif** (num2\_flag = false) **then**

digit2 := to\_integer(unsigned(SW(3 **downto** 0)));

HEX2 <= DigitDisplay(SW(3 **downto** 0),digit2);

digit3 := to\_integer(unsigned(SW(7 **downto** 4)));

HEX3 <= DigitDisplay(SW(7 **downto** 4),digit3);

-- Check num2

err\_num2 := DigitCheck(digit3, digit2);

-- Turn off HEX0 and

HEX1 <= "1111111";

HEX0 <= "1111111";

**end** **if**;

**else**

-- Add up 2 number then display

result := num1 + num2;

hundred := result / 100;

tenth := (result / 10) **mod** 10;

unit := result **mod** 10;

**if** (result >= 100) **then**

HEX3 <= "1111111";

HEX2 <= SevenSegmentDisplay(std\_logic\_vector(to\_unsigned(hundred,4)));

HEX1 <= SevenSegmentDisplay(std\_logic\_vector(to\_unsigned(tenth,4)));

HEX0 <= SevenSegmentDisplay(std\_logic\_vector(to\_unsigned(unit,4)));

**elsif** result >= 10 **then**

HEX3 <= "1111111";

HEX2 <= "1111111";

HEX1 <= SevenSegmentDisplay(std\_logic\_vector(to\_unsigned(tenth,4)));

HEX0 <= SevenSegmentDisplay(std\_logic\_vector(to\_unsigned(unit,4)));

**else**

HEX3 <= "1111111";

HEX2 <= "1111111";

HEX1 <= "1111111";

HEX0 <= SevenSegmentDisplay(std\_logic\_vector(to\_unsigned(unit,4)));

**end** **if**;

-- Check if error

**if** (err\_num1 **or** err\_num2) **then**

HEX3 <= "1111111";

HEX2 <= "0000110"; --E

HEX1 <= "0101111"; --r

HEX0 <= "0101111"; --r

**end** **if**;

**end** **if**;

**end** **if**;

**end** **process** Control\_Number;

-- BT0 to confirm number

Check\_BT0: **process** (BT(0),BT(3))

**begin**

**if** (BT(3) = '0') **then**

LED (3 **downto** 0) <= "0000";

num1\_flag := false;

**end** **if**;

**if** (BT(0) = '0') **then**

**if** (num1\_flag = false) **then**

num1\_flag := true;

**if** **not** err\_num1 **then**

num1 := digit1 \* 10 + digit0;

**else**

LED (3 **downto** 0) <= "1111";

**end** **if**;

**end** **if**;

**end** **if**;

**end** **process** Check\_BT0;

-- BT1 to confirm number 2

Check\_BT1: **process** (BT(1),BT(3))

**begin**

**if** (BT(3) = '0') **then**

LED (7 **downto** 4) <= "0000";

num2\_flag := false;

**end** **if**;

**if** (BT(1) = '0') **then**

**if** (num2\_flag = false) **then**

num2\_flag := true;

**if** **not** err\_num2 **then**

num2 := digit3 \* 10 + digit2;

**else**

LED (7 **downto** 4) <= "1111";

**end** **if**;

**end** **if**;

**end** **if**;

**end** **process** Check\_BT1;

-- Check BT2

Calculate\_BT2: **process** (BT(2), BT(3))

**begin**

**if** BT(3) = '0' **then**

add\_flag := false;

**end** **if**;

**if** BT(2) = '0' **then**

add\_flag := true;

**end** **if**;

**end** **process** Calculate\_BT2;

**end** behavioral;

## Verilog code

**module MTran\_Lab4\_Verilog\_7SegmentLed** (

**input** [9:0] SW,

input [3:0] BT,

output reg [7:0] LED,

output reg [6:0] HEX0,

output reg [6:0] HEX1,

output reg [6:0] HEX2,

output reg [6:0] HEX3

)**;**

// Variable declaration

**reg** add\_flag, num1\_flag, num2\_flag**;**

integer hundred, tenth, unit;

integer err, err\_num1, err\_num2, digit0, digit1, digit2, digit3, num1, num2, result;

initial

**begin**

add\_flag = 0;

num1\_flag = 0;

num2\_flag = 0;

hundred = 0;

tenth = 0;

unit = 0;

err = 0;

err\_num1 = 0;

err\_num2 = 0;

digit0 = 0;

digit1 = 0;

digit2 = 0;

digit3 = 0;

num1 = 0;

num2 = 0;

result = 0;

**end**

// Function Declaration

// Displaying 7 segment function

function [6:0] SevenSegmentDisplay;

**input** [3:0] Switch**;**

**begin**

**case** **(**Switch**)**

4'b0000**:** SevenSegmentDisplay = 7'b1000000**;**

4'b0001**:** SevenSegmentDisplay = 7'b1111001**;**

4'b0010**:** SevenSegmentDisplay = 7'b0100100**;**

4'b0011**:** SevenSegmentDisplay = 7'b0110000**;**

4'b0100**:** SevenSegmentDisplay = 7'b0011001**;**

4'b0101**:** SevenSegmentDisplay = 7'b0010010**;**

4'b0110**:** SevenSegmentDisplay = 7'b0000010**;**

4'b0111**:** SevenSegmentDisplay = 7'b1111000**;**

4'b1000**:** SevenSegmentDisplay = 7'b0000000**;**

4'b1001**:** SevenSegmentDisplay = 7'b0010000**;**

4'b1010**:** SevenSegmentDisplay = 7'b0001000**;**

4'b1011**:** SevenSegmentDisplay = 7'b0000011**;**

4'b1100**:** SevenSegmentDisplay = 7'b1000110**;**

4'b1101**:** SevenSegmentDisplay = 7'b0100001**;**

4'b1110**:** SevenSegmentDisplay = 7'b0000110**;**

4'b1111**:** SevenSegmentDisplay = 7'b0001110**;**

**default:** SevenSegmentDisplay = 7'b1111111**;**

**endcase**

**end**

endfunction

// Function to display a digit on 7-segment display

function [6:0] DigitDisplay;

**input** [3:0] Switch**;**

**input** integer dig**;**

**begin**

**if** (dig <= 9)

**begin**

DigitDisplay = SevenSegmentDisplay(Switch);

**end**

**else**

**begin**

err = err + 1;

DigitDisplay = SevenSegmentDisplay(4'b1111);

**end**

**end**

endfunction

// Main code

//

**always** @(SW)

**begin**

// Reset

err = 0;

// SW9 to change different assignment

**if** (SW[9] == 1'b0)

**begin**

HEX3 = 7'b1111111;

HEX2 = 7'b1111111;

HEX1 = 7'b1111111;

HEX0 = SevenSegmentDisplay(SW[3:0]);

**end**

**else**

**begin**

// Displaying number 1 and number 2

**if** (!add\_flag)

**begin**

**if** (!num1\_flag)

**begin**

digit0 = SW[3:0];

HEX0 = DigitDisplay(SW[3:0], digit0);

digit1 = SW[7:4];

HEX1 = DigitDisplay(SW[7:4], digit1);

// Turn off HEX2 and HEX3

HEX2 = 7'b1111111;

HEX3 = 7'b1111111;

**end**

**else**

**begin**

digit2 = SW[3:0];

HEX2 = DigitDisplay(SW[3:0], digit2);

digit3 = SW[7:4];

HEX3 = DigitDisplay(SW[7:4], digit3);

// Turn off HEX0 and HEX1

HEX1 = 7'b1111111;

HEX0 = 7'b1111111;

**end**

**end**

**else**

// Add up 2 number then display

**begin**

result = num1 + num2;

hundred = result / 100;

tenth = (result / 10) % 10;

unit = result % 10;

**if** (result >= 100)

**begin**

HEX3 = 7'b1111111;

HEX2 = SevenSegmentDisplay(hundred);

HEX1 = SevenSegmentDisplay(tenth);

HEX0 = SevenSegmentDisplay(unit);

**end**

**else** **if** (result >= 10)

**begin**

HEX3 = 7'b1111111;

HEX2 = 7'b1111111;

HEX1 = SevenSegmentDisplay(tenth);

HEX0 = SevenSegmentDisplay(unit);

**end**

**else**

**begin**

HEX3 = 7'b1111111;

HEX2 = 7'b1111111;

HEX1 = 7'b1111111;

HEX0 = SevenSegmentDisplay(unit);

**end**

// Check if error

**if** (err\_num1 || err\_num2)

**begin**

HEX3 = 7'b1111111;

HEX2 = 7'b0000110; //E

HEX1 = 7'b0101111; //r

HEX0 = 7'b0101111; //r

**end**

**end**

**end**

**end**

// Check BT0

**always**@ (negedge(BT[0]) or negedge(BT[3]))

**begin**

// Reset

**if** (!BT[3])

**begin**

num1\_flag = 0;

err\_num1 = 0;

LED[3:0] = 4'b0000;

**end**

// Assigning num1

**if** (~BT[0])

**begin**

num1\_flag = 1;

**if** (err == 0)

**begin**

num1 = digit1 \* 10 + digit0;

**end**

**else**

**begin**

LED[3:0] = 4'b1111;

err\_num1 = 1;

**end**

**end**

**end**

// Check BT1

**always**@ (negedge(BT[1]) or negedge(BT[3]))

**begin**

// Reset

**if** (!BT[3])

**begin**

num2\_flag = 0;

err\_num2 = 0;

LED[7:4] = 4'b0000;

**end**

// Assigning num2

**if** (!BT[1])

**begin**

num2\_flag = 1;

**if** (err == 0)

**begin**

num2 = digit3 \* 10 + digit2;

**end**

**else**

**begin**

LED[7:4] = 4'b1111;

err\_num2 = 1;

**end**

**end**

**end**

**always**@ (negedge(BT[2]) or negedge(BT[3]))

**begin**

**if** (!BT[3])

**begin**

add\_flag = 0;

**end**

**else**

**begin**

add\_flag = 1;

**end**

**end**

endmodule

# Pin Planner

## Input and Output

A close-up of a circuit board

Description automatically generated

Figure 4‑1: SCEMA5F31C6N board

Assigning:

BT[0]: KEY0

BT[1]: KEY1

BT[2]: KEY2

SW[0]: SW0

SW[1]: SW1

SW[2]: SW2

SW[3]: SW3

SW[4]: SW4

SW[5]: SW5

SW[6]: SW6

SW[7]: SW7

SW[8]: SW8

SW[9]: SW9

LED[0]: LEDR0

LED[1]: LEDR1

LED[2]: LEDR2

LED[3]: LEDR3

LED[4]: LEDR4

LED[5]: LEDR5

LED[6]: LEDR6

LED[7]: LEDR7

|  |  |  |  |
| --- | --- | --- | --- |
| HEX0[0]: HEX0[0]  HEX0[1]: HEX0[1]  HEX0[2]: HEX0[2]  HEX0[3]: HEX0[3]  HEX0[4]: HEX0[4]  HEX0[5]: HEX0[5]  HEX0[6]: HEX0[6] | HEX1[0]: HEX1[0]  HEX1[1]: HEX1[1]  HEX1[2]: HEX1[2]  HEX1[3]: HEX1[3]  HEX1[4]: HEX1[4]  HEX1[5]: HEX1[5]  HEX1[6]: HEX1[6] | HEX2[0]: HEX2[0]  HEX2[1]: HEX2[1]  HEX2[2]: HEX2[2]  HEX2[3]: HEX2[3]  HEX2[4]: HEX2[4]  HEX2[5]: HEX2[5]  HEX2[6]: HEX2[6] | HEX3[0]: HEX3[0]  HEX3[1]: HEX3[1]  HEX3[2]: HEX3[2]  HEX3[3]: HEX3[3]  HEX3[4]: HEX3[4]  HEX3[5]: HEX3[5]  HEX3[6]: HEX3[6] |

From the DE1\_SoC\_User\_Manual,

A screenshot of a slide switch

Description automatically generated

Figure 4‑2: SW0 and SW1 Pin No

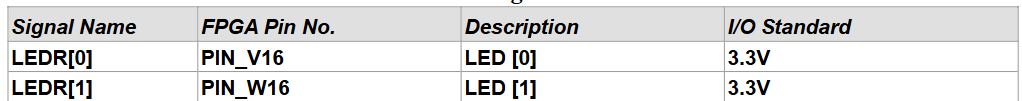


Figure 4‑3: LEDR0's Pin No

 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.

SW[3:0] all equal to 0 in which displaying 0.

A close up of a circuit board

Description automatically generated

Figure 5‑2: When 1 level is full

SW3 = 1, SW2 = 0, SW1 = 0, SW0 = 1 which should be 9 and the 7 segment LED displaying 9.

A close up of a circuit board

Description automatically generated

Figure 5‑3: When all levels are full

SW3 = 1, SW2 = 1, SW1 = 1, SW0 = 0 which should be 14 and the 7 segment LED displaying ‘E’.

**Calculator:**

A close up of a circuit board

Description automatically generated

Figure 5‑4: Choosing number 1

Input are SW[7:0] to choose number 1. SW[7:4] controlled HEX1 and SW[3:0] controlled HEX0.

Input BT0 is used to confirm the first number.

A close up of a circuit board

Description automatically generated

Figure 5‑5: Choosing number 2

Also input are SW[7:0] to choose number 2. SW[7:4] controlled HEX3 and SW[3:0] controlled HEX2.

Input BT1 is used to confirm the number.

A close up of a circuit board

Description automatically generated

Figure 5‑6: Result for adding them up

After confirming both number, press button 2 to add both number and HEX[2:0] will display the result.

**REFERENCES**